

# MAJOR PROJECT

*Submitted in partial fulfillment of the  
requirements for the award of the degree*

of

**Bachelor of Technology**

in

**Computer Science & Engineering**

By

**Ashish Batra (40176802717/CSE-3/2017)**  
**Priyanshi Verma (03576802717/CSE-3/2017)**  
**Sauransh Nayyar (40476802717/CSE-3/2017)**  
**Dhruv Chawla (50376802717/CSE-3/2017)**



**Department of Computer Science & Engineering**  
**Guru Tegh Bahadur Institute of Technology**

**Guru Gobind Singh Indraprastha University**  
**Dwarka, New Delhi**  
**Year 2017-2021**

# **UBER DATA ANALYSIS USING R STUDIO**



By

**Ashish Batra (40176802717/CSE-3/2017)**  
**Priyanshi Verma (03576802717/CSE-3/2017)**  
**Sauransh Nayyar (40476802717/CSE-3/2017)**  
**Dhruv Chawla (50376802717/CSE-3/2017)**

## DECLARATION

We hereby declare that all the work presented in this **Major Report** for the partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering**, **Guru Tegh Bahadur Institute of Technology**, affiliated to **Guru Gobind Singh Indraprastha University Delhi** is an authentic record of our own work.

**Date: 13th July, 2021**

**Ashish Batra (40176802717/CSE-3/2017)**  
**Priyanshi Verma (03576802717/CSE-3/2017)**  
**Sauransh Nayyar (40476802717/CSE-3/2017)**  
**Dhruv Chawla (50376802717/CSE-3/2017)**



## ACKNOWLEDGEMENT

We express our deep sense of gratitude and obligations to our mentor, **Dr. Munshi Yadav** who provided us with their expert guidance, support, and suggestions about the project. Without their help, we wouldn't have been able to present this piece of work to the present standard. We also take this opportunity to give thanks to all the other people who gave us support for the project or other aspects of our study allowing us to perform to the best of our abilities.

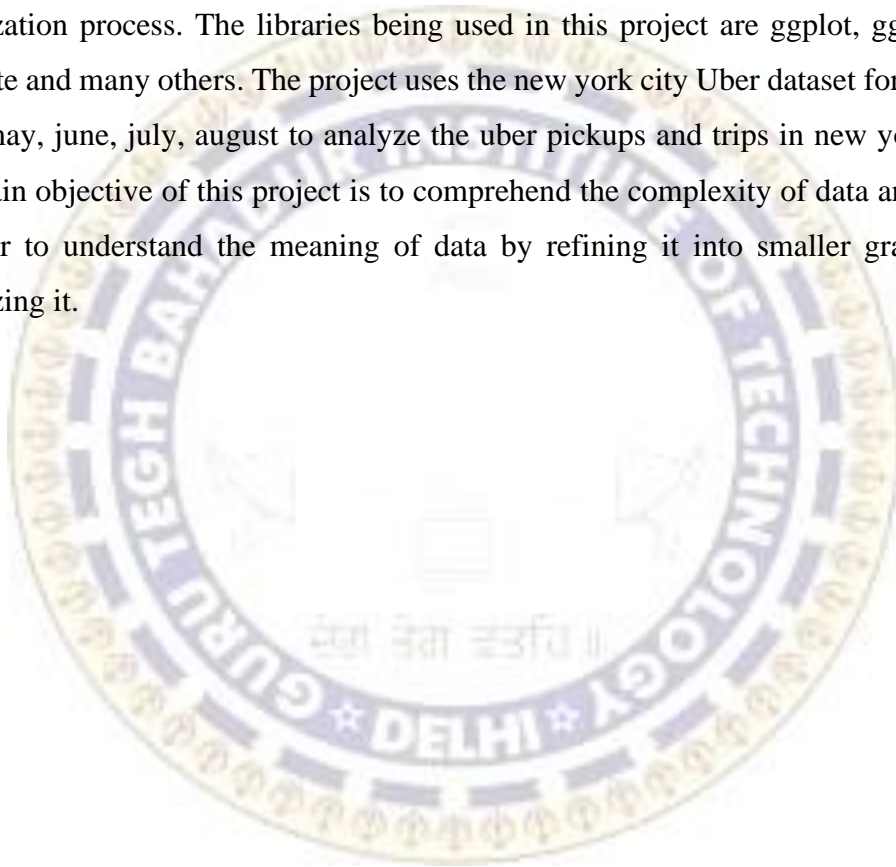
Date: 13th July, 2021

Ashish Batra (40176802717/CSE-3/2017)  
Priyanshi Verma (03576802717/CSE-3/2017)  
Sauransh Nayyar (40476802717/CSE-3/2017)  
Dhruv Chawla (50376802717/CSE-3/2017)



## ABSTRACT

The project titled 'Uber Data Analysis Project' emphasizes on data visualization techniques and the various ways to visualize data according to the requirements. Data visualization is the graphical representation of information and data. With the help of visualization, it becomes feasible to gain the benefit of understanding the complex data and insights that would help individuals to craft decisions. Here, we have made use of R as our programming language and Rstudio as a platform for implementing the data visualization process. The libraries being used in this project are ggplot, ggthemes, lubridate and many others. The project uses the new york city Uber dataset for months april, may, june, july, august to analyze the uber pickups and trips in new york city. The main objective of this project is to comprehend the complexity of data and make it easier to understand the meaning of data by refining it into smaller grains and visualizing it.



## TABLES AND FIGURES

<b>Fig No</b>	<b>Figure Name</b>	<b>Page</b>
1.1	Sample bar graph	4
1.2	Sample histogram	4
1.3	Sample line graph	5
1.4	Sample Pie Chart	5
1.5	Sample Scatter Plot	6
2.1	Box Plot Example	11
2.2	Sample Box Plots with different themes	11
2.3	Sample Box Plots with theme changes	12
2.4	Sample Box Plots with theme changes	12
2.5	Sample Box Plots with theme changes	12
2.6	Sample Box Plots with theme changes	13
4.1	Histogram Example	22
4.2	Line Chart Example	23
4.3	Stacked Bar Graph Example	24
4.4	Box Plot Example	25
4.5	Scatter plot Example	26
4.6	3D Scatter plot Example	26
4.7	Hexagon Binding Example	27
4.8	Coloured Hexagon Binding Plot Example	28
4.9	Mosaic plot Example	28
4.10	Heat Map Example	29
4.11	Map Visualization Example	30
4.12	3D graphs	30
4.13	3D Scatter plot graph	31
4.14	Correlogram (GUIs) Example	32

## CONTENTS

Chapter	Page No.
Title Page	
Declaration	i
Acknowledgement	ii
Abstract	iii
Tables and figures	iv
1. Data Visualization	1
1.1. Data Visualization Techniques	3-6
2. Introduction to R	7
2.1. The R Environment	8
2.2. R Libraries used in project	9-17
3. R Studio	18-19
4. Data Visualization in R Studio	20-32
5. Uber Data Analysis	33-38
Summary and Conclusion	40
Appendix A (Source Code)	41-46
Appendix B (Screenshots)	47-59
References	60-63





# 1. Data Visualization

Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data.

In the world of big data, data visualization tools and technologies are essential to analyse massive amounts of information and make data-driven decisions.

From an academic point of view, this representation can be considered as a mapping between the original data (usually numerical) and graphic elements (for example, lines or points in a chart). The mapping determines how the attributes of these elements vary according to the data. In this light, a bar chart is a mapping of the length of a bar to a magnitude of a variable. Since the graphic design of the mapping can adversely affect the readability of a chart, mapping is a core competency of data visualization.

Data visualization has its roots in the field of Statistics and is therefore generally considered a branch of descriptive statistics. However, because both design skills and statistical and computing skills are required to visualize effectively, it is argued by some authors that it is both an art and a science.

To communicate information clearly and efficiently, data visualization uses statistical graphics, plots, information graphics and other tools. Numerical data may be encoded using dots, lines, or bars, to visually communicate a quantitative message. Effective visualization helps users analyse and reason about data and evidence. It makes complex data more accessible, understandable, and usable. Users may have particular analytical tasks, such as making comparisons or understanding causality, and the design principle of the graphic (i.e., showing comparisons or showing causality) follows the task. Tables are generally used where users will look up a specific measurement, while charts of various types are used to show patterns or relationships in the data for one or more variables.

Data visualization refers to the techniques used to communicate data or information by encoding it as visual objects (e.g., points, lines, or bars) contained in graphics. The goal is to communicate information clearly and efficiently to users. It is one of the steps in data analysis or data science.

Data visualization is an increasingly used key tool to make sense of the trillions of rows of data generated every day. It helps to tell stories by curating data into a form easier to understand, highlighting the trends and outliers. A good visualization tells a story, removing the noise from data and highlighting the useful information.

Effective data visualization is a delicate balancing act between form and function. The plainest graph could be too boring to catch any notice, or it makes a powerful point;

the most stunning visualization could utterly fail at conveying the right message or it could speak volumes. The data and the visuals need to work together, and there's an art to combining great analysis with great storytelling.

Data visualization involves specific terminology, some of which is derived from statistics. For example, author Stephen Few defines two types of data, which are used in combination to support a meaningful analysis or visualization:

- **Categorical:** Represent groups of objects with a characteristic. Categorical variables can either be nominal or ordinal. Nominal variables for example gender have no order between them and are thus nominal. Ordinal variables are categories with an order, for sample recording the age group someone falls into.
- **Quantitative:** Represent measurements, such as the height of a person or the temperature of an environment. Quantitative variables can either be continuous or discrete. Continuous variables capture the idea that measurements can always be made more precisely. Discrete variables have only a finite number of possibilities, such as a count of some outcomes or an age measured in whole years.

The distinction between quantitative and categorical variables is important because the two types require different methods of visualization.

Two primary types of information displays are tables and graphs.

- A table contains quantitative data organized into rows and columns with categorical labels. It is primarily used to look up specific values. In the example above, the table might have categorical column labels representing the name (a qualitative variable) and age (a quantitative variable), with each row of data representing one person (the sampled experimental unit or category subdivision).
- A graph is primarily used to show relationships among data and portrays values encoded as visual objects (e.g., lines, bars, or points). Numerical values are displayed within an area delineated by one or more axes. These axes provide scales (quantitative and categorical) used to label and assign values to the visual objects. Many graphs are also referred to as charts.

Visualization is the first step to make sense of data. To translate and present data and data correlations in a simple way, data analysts use a wide range of techniques — charts, diagrams, maps, etc. Choosing the right technique and its setup is often the only way to make data understandable. Vice versa, poorly selected tactics won't unlock the full potential of data or even make it irrelevant.

The factors that influence data visualization choices:

1. **Audience.** It's important to adjust data representation to the specific target audience. For example, fitness mobile app users who browse through their progress can easily work with uncomplicated visualizations. On the other hand, if data insights are intended for researchers and experienced decision-makers who regularly work with data, you can and often have to go beyond simple charts.
2. **Content.** The type of data you are dealing with will determine the tactics. For example, if it's time-series metrics, you will use line charts to show the dynamics in many cases. To show the relationship between two elements, scatter plots are often used. In turn, bar charts work well for comparative analysis.
3. **Context.** You can use different data visualization approaches and read data depending on the context. To emphasize a certain figure, for example, significant profit growth, you can use the shades of one color on the chart and highlight the highest value with the brightest one. On the contrary, to differentiate elements, you can use contrast colors.
4. **Dynamics.** There are various types of data, and each type has a different rate of change. For example, financial results can be measured monthly or yearly, while time series and tracking data are changing constantly. Depending on the rate of change, you may consider dynamic representation (steaming) or static visualization techniques in data mining.
5. **Purpose.** The goal of data visualization affects the way it is implemented. In order to make a complex analysis, visualizations are compiled into dynamic and controllable dashboards that work as visual data analysis techniques and tools. However, dashboards are not necessary to show a single or occasional data insight.

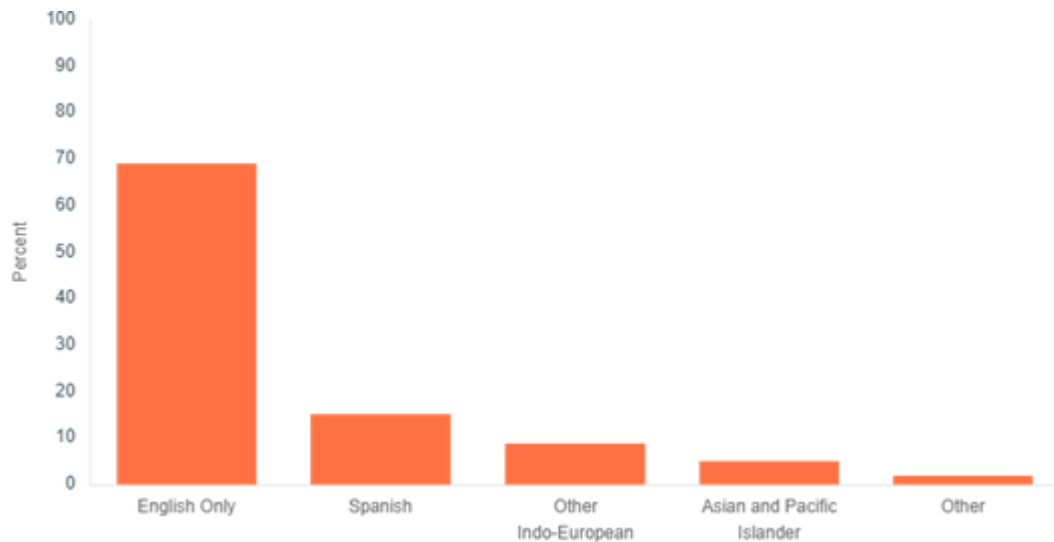
## 1.1 Data visualization techniques

### (A) Bar Chart

A bar chart displays categorical data with rectangular bars whose length or height corresponds to the value of each data point.

Bar charts can be visualized using vertical or horizontal bars. Bar charts are best used to compare a single category of data or several. When comparing more than one category of data, the bars can be grouped together to create a grouped bar chart.

Bar charts use volume to demonstrate differences between each bar. Because of this, bar charts should always start at zero. When bar charts do not start at zero, it risks users misjudging the difference between data values

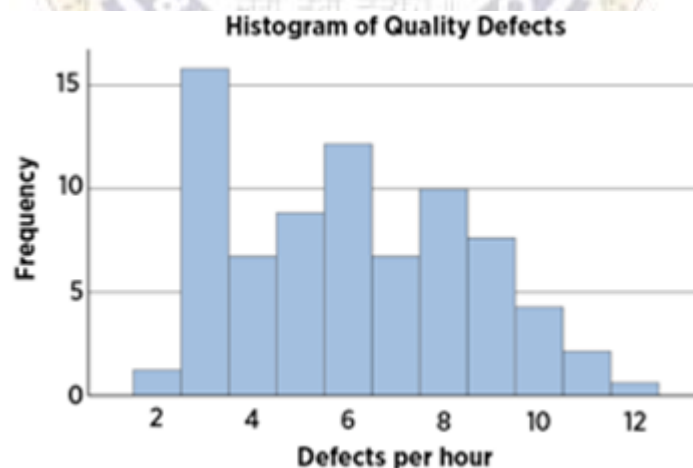


**fig1.1 Sample bar graph**

### **(B) Histogram**

A histogram is a chart that displays numeric data in ranges, where each bar represents how frequently numbers fall into a range.

Like a bar chart, histograms consist of a series of vertical bars along the x-axis. Histograms are mostly used to depict what a set of data looks like in aggregate. At a quick glance, histograms tell whether a dataset has values that are clustered around a small number of ranges or are more spread out.



**fig 1.2 Sample histogram**

### **(C) Line Graph**

A line graph is a chart that displays a series of data points connected by line segments. Line graphs can be used to show how data changes over time and are often used to communicate trends, such as how household income changes each year. The x-axis

can display continuous or discrete data such as days, years, or categories, and the y-axis is typically a continuous variable but can also be discrete. Line graphs can also be used to compare multiple trend lines, whereas bar charts can only represent one trend line.

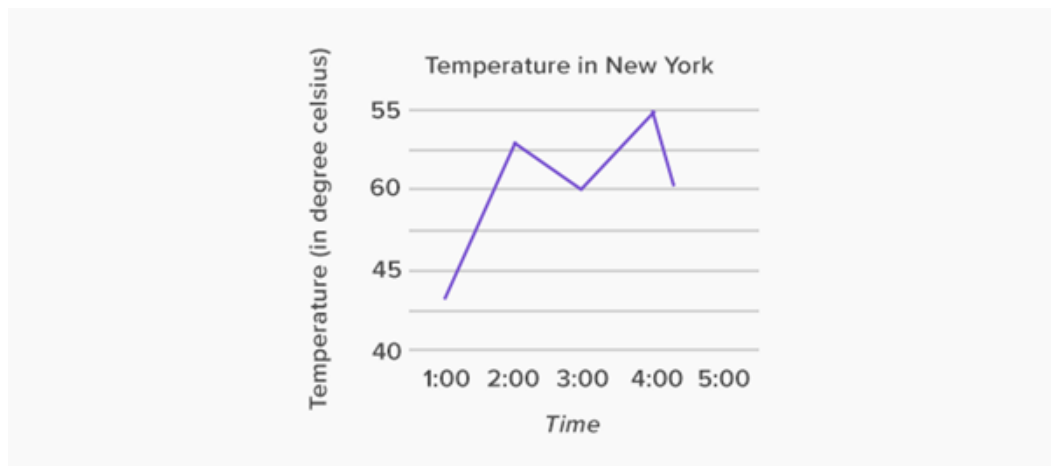


fig 1.3 Sample line graph

#### (D) Pie Chart

A pie chart is a circular graph that presents values as proportionate slices. Pie charts are one of the most commonly used data visualizations but are often not the most effective way to compare data values. This is mainly due to the size of each slice and the number of data categories being represented. Pie charts can also be represented as a doughnut chart, which is generally better for users to compare the size of each slice or arc.

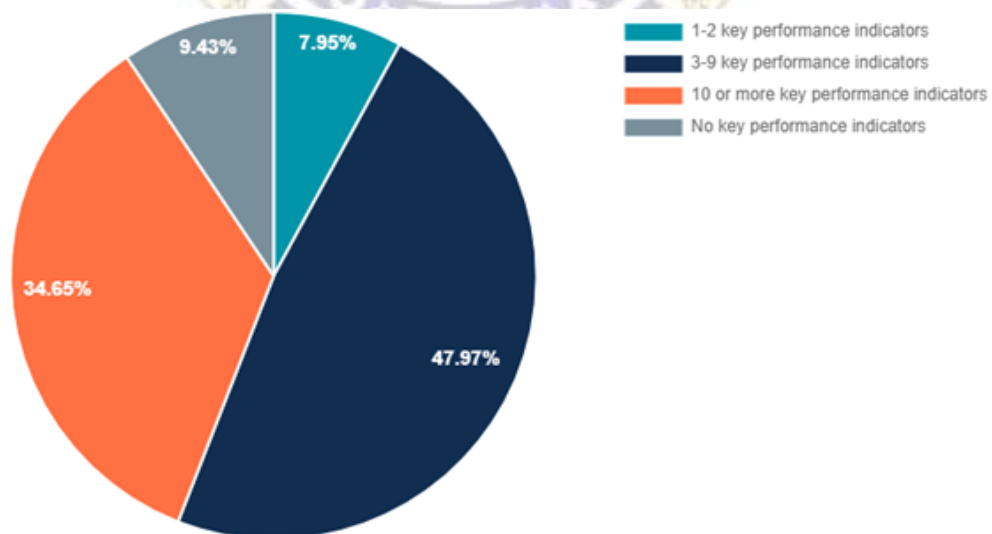


fig 1.4 Sample Pie Chart

### (E) Scatter Plot

A scatter plot is a two-dimensional chart that shows the relationship between two variables.

To do this, a series of values are plotted on an x-axis and a y-axis, with each axis representing a variable and each value representing a dot. Scatter plots often include an explanatory variable, such as years of education, and what might be considered a response variable, such as annual income. When the dots are plotted against these two axes, scatter plots communicate the strength and type of relationship that exists between these variables.

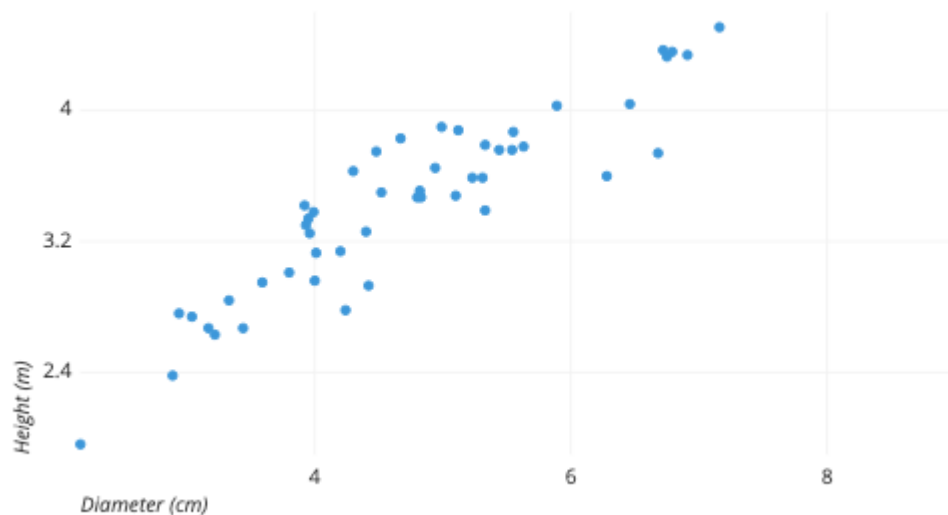
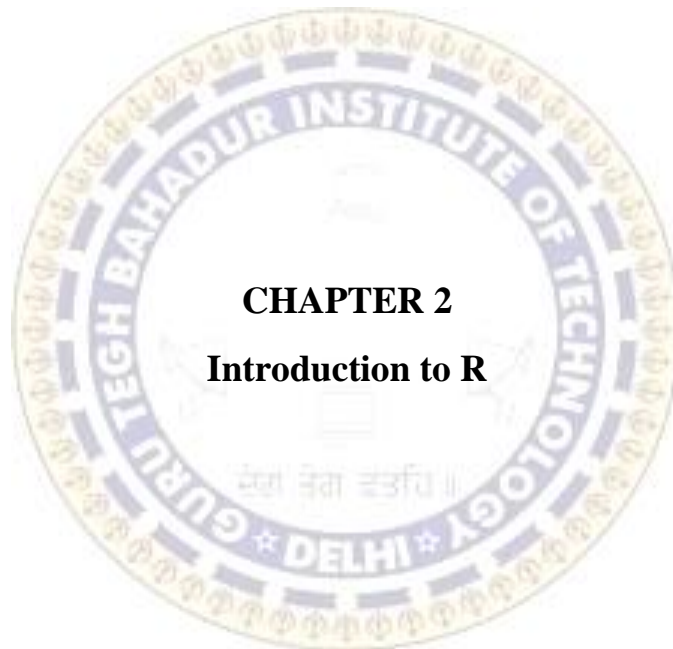


fig 1.5 Sample Scatter Plot





## 2. Introduction to R

R is a language and environment for statistical computing and graphics. It is a GNU project which is similar to the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues. R can be considered as a different implementation of S. There are some important differences, but much code written for S runs unaltered under R.

R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, ...) and graphical techniques, and is highly extensible. The S language is often the vehicle of choice for research in statistical methodology, and R provides an Open-Source route to participation in that activity.

One of R's strengths is the ease with which well-designed publication-quality plots can be produced, including mathematical symbols and formulae where needed. Great care has been taken over the defaults for the minor design choices in graphics, but the user retains full control.

R is available as Free Software under the terms of the Free Software Foundation's GNU General Public License in source code form. It compiles and runs on a wide variety of UNIX platforms and similar systems (including FreeBSD and Linux), Windows and MacOS.

### 2.1 The R environment

R is an integrated suite of software facilities for data manipulation, calculation and graphical display. It includes

- an effective data handling and storage facility,
- a suite of operators for calculations on arrays, in particular matrices,
- a large, coherent, integrated collection of intermediate tools for data analysis,
- graphical facilities for data analysis and display either on-screen or on hardcopy, and
- a well-developed, simple and effective programming language which includes conditionals, loops, user-defined recursive functions and input and output facilities.

The term “environment” is intended to characterize it as a fully planned and coherent system, rather than an incremental accretion of very specific and inflexible tools, as is frequently the case with other data analysis software.

R, like S, is designed around a true computer language, and it allows users to add additional functionality by defining new functions. Much of the system is itself written in the R dialect of S, which makes it easy for users to follow the algorithmic choices



made. For computationally intensive tasks, C, C++ and Fortran code can be linked and called at run time. Advanced users can write C code to manipulate R objects directly.

Many users think of R as a statistics system. We prefer to think of it as an environment within which statistical techniques are implemented. R can be extended (easily) via packages. There are about eight packages supplied with the R distribution and many more are available through the CRAN family of Internet sites covering a very wide range of modern statistics.

R has its own LaTeX-like documentation format, which is used to supply comprehensive documentation, both on-line in a number of formats and in hardcopy

## 2.2 R Libraries used in project

### (A) `library(ggplot2)`

ggplot2 is an open-source data visualization package for the statistical programming language R. Created by Hadley Wickham in 2005, ggplot2 is an implementation of Leland Wilkinson's Grammar of Graphics—a general scheme for data visualization which breaks up graphs into semantic components such as scales and layers. ggplot2 can serve as a replacement for the base graphics in R and contains a number of defaults for web and print display of common scales. Since 2005, ggplot2 has grown in use to become one of the most popular R packages.

### **ggplot2 package**

**ggplot2** is a powerful and flexible **R package**, implemented by **Hadley Wickham**, for producing elegant graphics.

The concept behind ggplot2 divides plot into three different fundamental parts: **Plot = data + Aesthetics + Geometry**.

The principal components of every plot can be defined as follow:

- **data** is a data frame
- **Aesthetics** is used to indicate x and y variables. It can also be used to control the **color**, the **size** or the **shape** of points, the height of bars, etc.....
- **Geometry** defines the type of graphics (**histogram**, **box plot**, **line plot**, **density plot**, **dot plot**, ....)

`install.packages("ggplot2")`

`library(ggplot2)`

There are two major functions in the ggplot2 package: **qplot()** and **ggplot()** functions.

- **qplot()** stands for quick plot, which can be used to easily produce simple plots.
- **ggplot()** function is more flexible and robust than **qplot** for building a plot piece by piece

While **qplot()** is useful to plot quickly, most of time, one should use **ggplot()** for systemic plotting

```
qplot(x, y, dataframe, geom="type")
ggplot(data, aes(x,y))+ geom_type() + options
x<-1:30
y<-c(5,4,5,3,4,6,7,4,2,1,5,6,2,2,2,
      5,6,7,8,6,2,8,7,8,8,3,3,7,2,1)
z<-c('A','B','A','A','B','A','B','B','A','A',
      'B','A','A','A','B','A','A','B','A','B',
      'B','A','A','A','B','A','A','B','A','B')
df<-data.frame(x,y,z)
```

### (B) library(ggthemes)

ggthemes is an open source data visualization library that has quick functions to change plot themes and to customize the appearance of the plot background. Using ggthemes, we can:

- o Change the colors of the plot panel background and the grid lines
- o Remove plot panel borders and grid lines
- o Change the plot background color (not the panel)

Use a custom theme

- o theme\_tufte : a minimalist theme
- o theme\_economist : theme based on the plots in the economist magazine
- o theme\_stata: theme based on Stata graph schemes.
- o theme\_wsj: theme based on plots in the Wall Street Journal
- o theme\_calc : theme based on LibreOffice Calc
- o theme\_hc : theme based on Highcharts JS
- o Functions: **theme()**, **theme\_bw()**, **theme\_grey()**, **theme\_update()**, **theme\_blank()**, **theme\_classic()**, **theme\_minimal()**, **theme\_void()**, **theme\_dark()**, **element\_blank()**, **element\_line()**, **element\_rect()**, **element\_text()**, **rel()**

### Example of plot

```
library(ggplot2)
p <- ggplot(ToothGrowth, aes(x=dose, y=len)) + geom_boxplot()
P
```

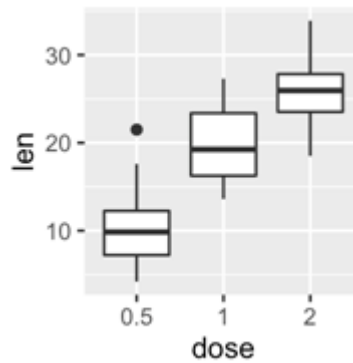


fig 2.1 Box Plot Example

### Quick functions to change plot themes

Several functions are available in ggplot2 package for changing quickly the theme of plots:

- **theme\_gray** : gray background color and white grid lines
- **theme\_bw** : white background and gray grid lines

`p + theme_gray(base_size = 14)`

`p + theme_bw()`

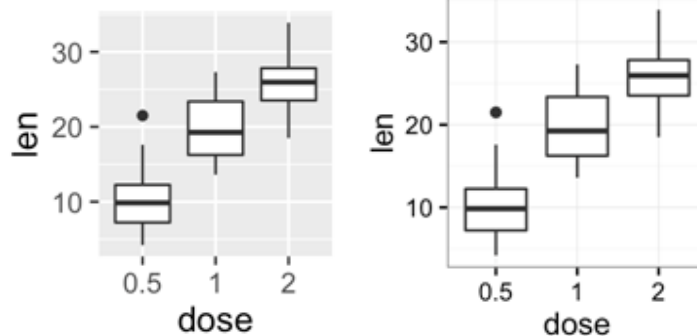


fig 2.2 Sample Box Plots with different themes

- **theme\_linedraw** : black lines around the plot
- **theme\_light** : light gray lines and axis (more attention towards the data)

`p + theme_linedraw()`

`p + theme_light()`

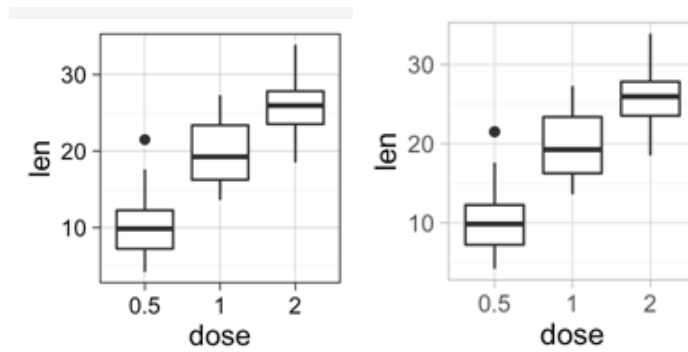


fig 2.3 Sample Box Plots with theme changes

- **theme\_minimal**: no background annotations
  - **theme\_classic** : theme with axis lines and no grid lines
- `p + theme_minimal()`  
`p + theme_classic()`

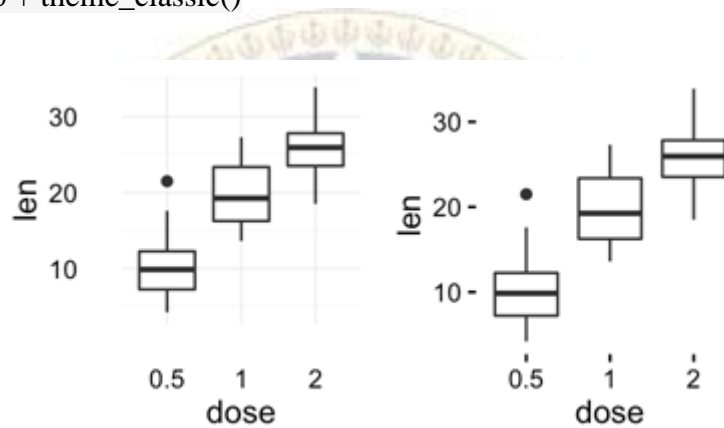


fig 2.4 Sample Box Plots with theme changes

- **theme\_void**: Empty theme, useful for plots with non-standard coordinates or for drawings
  - **theme\_dark**(): Dark background designed to make colours pop out
- `p + theme_void()`  
`p + theme_dark()`

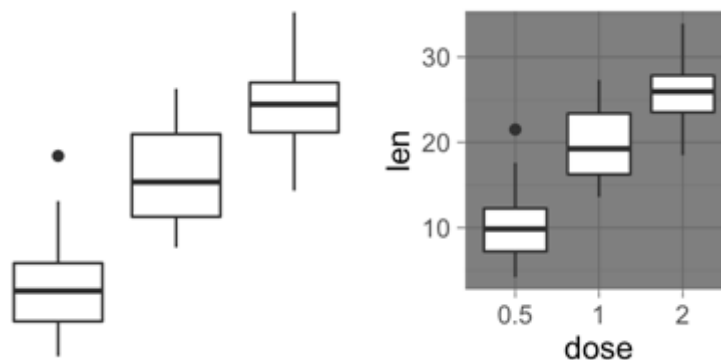


fig 2.5 Sample Box Plots with theme changes

The functions `theme_xx()` can take the two arguments below :

- **base\_size** : base font size (to change the size of all plot text elements)
- **base\_family** : base font family

The size of all the plot text elements can be easily changed at once :

# Example 1

```
theme_set(theme_gray(base_size = 20))
```

```
ggplot(ToothGrowth, aes(x=dose, y=len)) + geom_boxplot()
```

# Example 2

```
ggplot(ToothGrowth, aes(x=dose, y=len)) + geom_boxplot()+
```

```
theme_classic(base_size = 25)
```

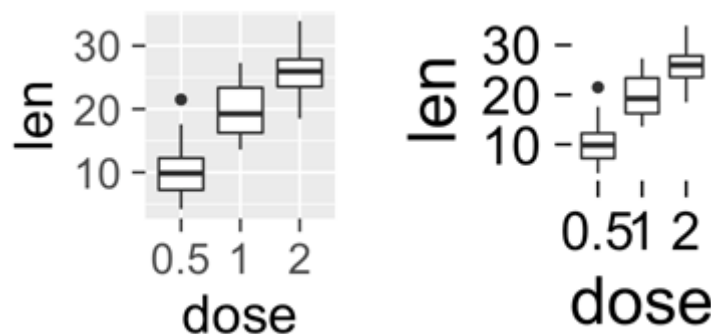


fig 2.6 Sample Box Plots with theme changes

### (C) **library(lubridate)**

Lubridate is an R package that makes it easier to work with dates and times. Below is a concise tour of some of the things lubridate can do for you. Lubridate was created by Garrett Grolemund and Hadley Wickham, and is now maintained by Vitalie Spinu.

Parsing dates and times

Getting R to agree that your data contains the dates and times you think it does can be tricky. Lubridate simplifies that. Identify the order in which the year, month, and day appears in your dates. Now arrange "y", "m", and "d" in the same order. This is the name of the function in lubridate that will parse your dates. For example,

```
library(lubridate)
```

```
#>
```

```
#> Attaching package: 'lubridate'
```

```
#> The following objects are masked from 'package:base':
```

```
#>
#>      date, intersect, setdiff, union
ymd("20110604")
#> [1] "2011-06-04"
mdy("06-04-2011")
#> [1] "2011-06-04"
dmy("04/06/2011")
#> [1] "2011-06-04"
```

Lubridate's parse functions handle a wide variety of formats and separators, which simplifies the parsing process.

If your date includes time information, add h, m, and/or s to the name of the function. `ymd_hms` is probably the most common date time format. To read the dates in a certain time zone, supply the official name of that time zone in the `tz` argument.

```
arrive <- ymd_hms("2011-06-04 12:00:00", tz = "Pacific/Auckland")
arrive
#> [1] "2011-06-04 12:00:00 NZST"
leave <- ymd_hms("2011-08-10 14:00:00", tz = "Pacific/Auckland")
leave
#> [1] "2011-08-10 14:00:00 NZST"
```

## Setting and Extracting information

Extract information from date times with the functions `second`, `minute`, `hour`, `day`, `wday`, `yday`, `week`, `month`, `year`, and `tz`. You can also use each of these to set (i.e, change) the given information. Notice that this will alter the date time. `wday` and `month` have an optional `label` argument, which replaces their numeric output with the name of the weekday or month.

```
second(arrive)
#> [1] 0
second(arrive) <- 25
arrive
```

```
#> [1] "2011-06-04 12:00:25 NZST"

second(arrive) <- 0

wday(arrive)

#> [1] 7

wday(arrive, label = TRUE)

#> [1] Sat

#> Levels: Sun < Mon < Tue < Wed < Thu < Fri < Sat
```

#### (D) library(dplyr)

The **dplyr** package in R is a structure of data manipulation that provides a uniform set of verbs, helping to resolve the most frequent data manipulation hurdles.

The dplyr package performs the steps given below quicker and in an easier fashion:

- o By limiting the choices the focus can now be more on data manipulation difficulties.
- o There are uncomplicated “verbs”, functions present for tackling every common data manipulation and the thoughts can be translated into code faster.
- o There are valuable backends and hence waiting time for the computer reduces.

#### Important Verb Functions

dplyr package provides various important functions that can be used for Data Manipulation. These are:

- o **filter() Function:** For choosing cases and using their values as a base for it.
- o **arrange():** For reordering of the cases.
- o **select() and rename():** For choosing variables and using their names as a base for doing so.
- o **mutate() and transmute():** Addition of new variables which are the functions of prevailing variables.
- o **summarise():** Condensing various values to one value.
- o **sample\_n() and sample\_frac():** For taking random specimens.

### (E) library(tidyr)

tidyr is a new package that makes it easy to “tidy” your data. Tidy data is data that’s easy to work with: it’s easy to munge (with dplyr), visualise (with ggplot2 or ggvis) and model (with R’s hundreds of modelling packages). The two most important properties of tidy data are:

- o Each column is a variable.
- o Each row is an observation.

Arranging your data in this way makes it easier to work with because you have a consistent way of referring to variables (as column names) and observations (as row indices). When using tidy data and tidy tools, you spend less time worrying about how to feed the output from one function into the input of another, and more time answering your questions about the data.

To tidy messy data, you first identify the variables in your dataset, then use the tools provided by tidyr to move them into columns. tidyr provides three main functions for tidying your messy data: gather(), separate() and spread().

gather() takes multiple columns, and gathers them into key-value pairs: it makes “wide” data longer. Other names for gather include melt (reshape2), pivot (spreadsheets) and fold (databases).

### (F) library(DT)

The R package **DT** provides an R interface to the JavaScript library **DataTables**. R data objects (matrices or data frames) can be displayed as tables on HTML pages, and **DataTables** provides filtering, pagination, sorting, and many other features in the tables.

The main function in this package is datatable(). It creates an HTML widget to display R data objects with **DataTables**.

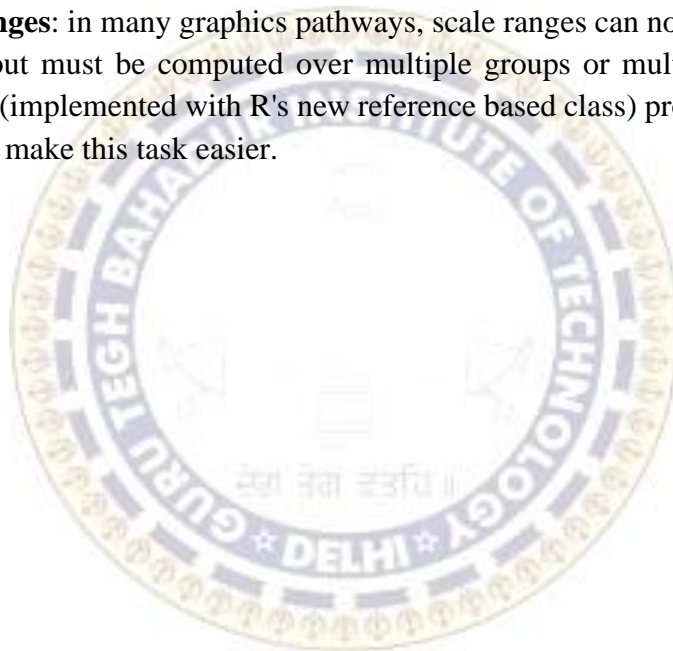
### (G) library(scales)

One of the most difficult parts of any graphics package is scaling, converting from data values to perceptual properties. The inverse of scaling, making guides (legends and axes) that can be used to read the graph, is often even harder! The idea of the scales package is to implement scales in a graphics system agnostic, so that everyone can benefit by pooling knowledge and resources about this tricky topic.



The scales package is made up of the following interdependent components

- o **Palettes:** pal for short, describe the useful palettes of aesthetics.
- o **Transformations:** trans for short, describe common scale transformations, their inverses, and ways of generating breaks and labels.
- o **Bounds:** various ways of rescaling the data
- o **Scaling functions:** pull together palettes, bounding functions and transformations to provide a complete pathway from raw data to perceptual properties
- o **Mutable ranges:** in many graphics pathways, scale ranges can not be computed in a single pass, but must be computed over multiple groups or multiple panels. The mutable ranges (implemented with R's new reference based class) provide a thin layer of mutability to make this task easier.





## **CHAPTER 3**

### **R Studio**

### 3. R studio

**RStudio** is an Integrated Development Environment (IDE) for R, a programming language for statistical computing and graphics. It is available in two formats: RStudio Desktop is a regular desktop application while RStudio Server runs on a remote server and allows accessing RStudio using a web browser.

RStudio Desktop and RStudio Server are both available in free and fee-based (commercial) editions. OS support depends on the format/edition of the IDE. Prepackaged distributions of RStudio Desktop are available for Windows, macOS, and Linux. RStudio Server and Server Pro run on Debian, Ubuntu, Red Hat Linux, CentOS, openSUSE and SLES.

The RStudio IDE is partly written in the C++ programming language and uses the Qt framework for its graphical user interface. The bigger percentage of the code is written in Java. JavaScript is also amongst the languages used.

Work on the RStudio IDE started around December 2010, and the first public beta version (v0.92) was officially announced in February 2011. Version 1.0 was released on 1 November 2016. Version 1.1 was released on 9 October 2017.

In April 2018, RStudio PBC (at the time RStudio, Inc.) announced that it will provide operational and infrastructure support to Ursa Labs in support of the Labs focus on building a new data science runtime powered by Apache Arrow.



The RStudio IDE is developed by RStudio Public-benefit corporation, a commercial enterprise founded by JJ Allaire, creator of the programming language ColdFusion. RStudio, PBC has no formal connection to the R Foundation, a not-for-profit organization located in Vienna, Austria, which is responsible for overseeing development of the R environment for statistical computing.



## **CHAPTER 4**

### **Data Visualization in R**

## 4. Data Visualization in R Studio

Historically, data visualization has evolved through the work of noted practitioners. The founder of graphical methods in statistics is William Playfair. William Playfair invented four types of graphs: the line graph, the bar chart of economic data, the pie chart and the circle graph. Joseph Priestly had created the innovation of the first timeline charts, in which individual bars were used to visualize the lifespan of a person (1765).

### Basic Visualization

1. Histogram
2. Bar / Line Chart
3. Box plot
4. Scatter plot

### Advanced Visualization

1. Heat Map
2. Mosaic Map
3. Map Visualization
4. 3D Graphs
5. Correlogram

### **Basic Visualizations**

1. Basic graphs in R can be created quite easily. The **plot** command is the command to note.
2. It takes in many parameters from x axis data, y axis data, x axis labels, y axis labels, color and title. To create line graphs, simply use the parameter, type=l.
3. If you want a boxplot, you can use the word boxplot, and for bar plot use the barplot function.

- **Histogram**

Histogram is basically a plot that breaks the data into bins (or breaks) and shows frequency distribution of these bins. You can change the breaks also and see the effect it has on data visualization in terms of understandability.

Note: We have used the par (mfrow=c(2,5)) command to fit multiple graphs in the same page for sake of clarity( see the code below).

The following commands show this in a better way. In the code below, the **main** option sets the Title of Graph and the **col** option calls in the color palette from RColorBrewer to set the colors.

```
library(RColorBrewer)
```

```
data(VADeaths)
```

```
par(mfrow=c(2,3))
```

```
hist(VADeaths,breaks=10, col=brewer.pal(3,"Set3"),main="Set3 3 colors")
```

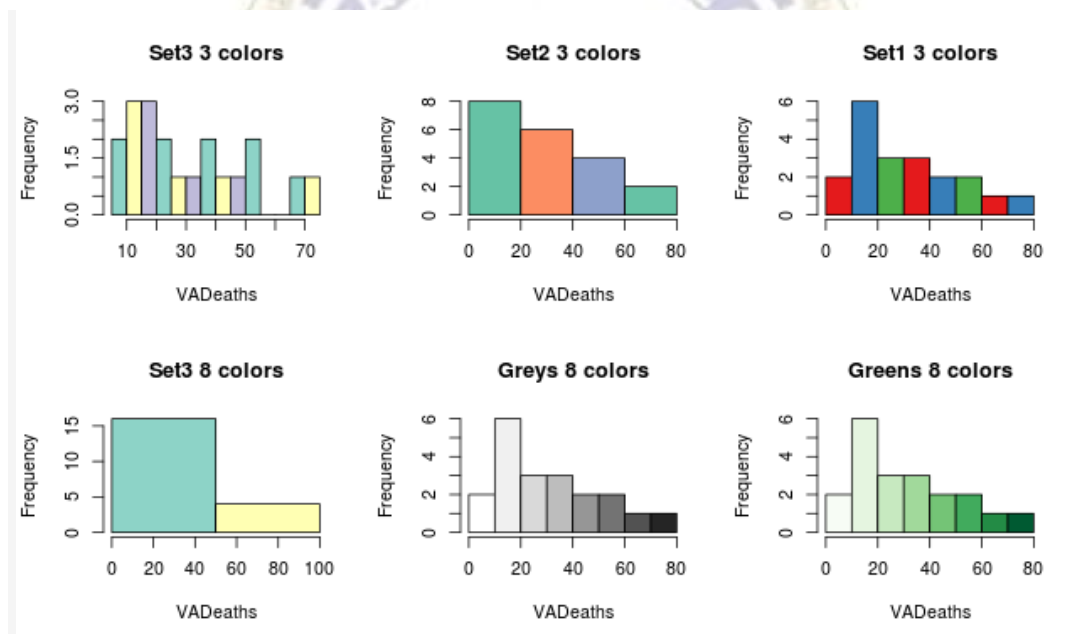
```
hist(VADeaths,breaks=3 ,col=brewer.pal(3,"Set2"),main="Set2 3 colors")
```

```
hist(VADeaths,breaks=7, col=brewer.pal(3,"Set1"),main="Set1 3 colors")
```

```
hist(VADeaths,.,breaks= 2, col=brewer.pal(8,"Set3"),main="Set3 8 colors")
```

```
hist(VADeaths,col=brewer.pal(8,"Greys"),main="Greys 8 colors")
```

```
hist(VADeaths,col=brewer.pal(8,"Greens"),main="Greens 8 colors")
```



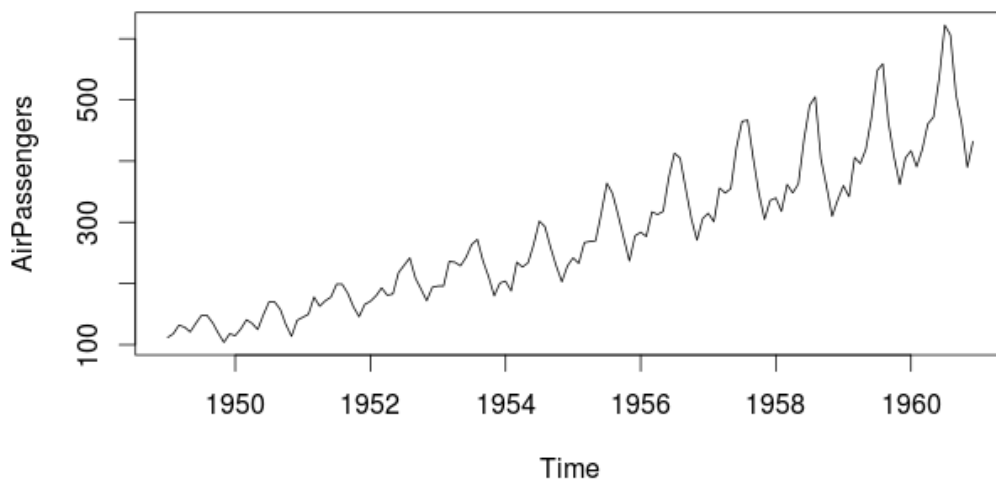
**fig 4.1 Histogram Example**

Notice, if the number of breaks is less than the number of colors specified, the colors just go to extreme values as in the “Set 3 8 colors” graph. If the number of breaks is more than the number of colors, the colors start repeating as in the first row.

- **Line Chart**

Below is the line chart showing the increase in air passengers over a given time period. Line Charts are commonly preferred when we are to analyse a trend spread over a time period. Furthermore, line plots are also suitable for plots where we need to compare relative changes in quantities across some variable (like time). Below is the code:

```
plot(AirPassengers,type="l") #Simple Line Plot
```



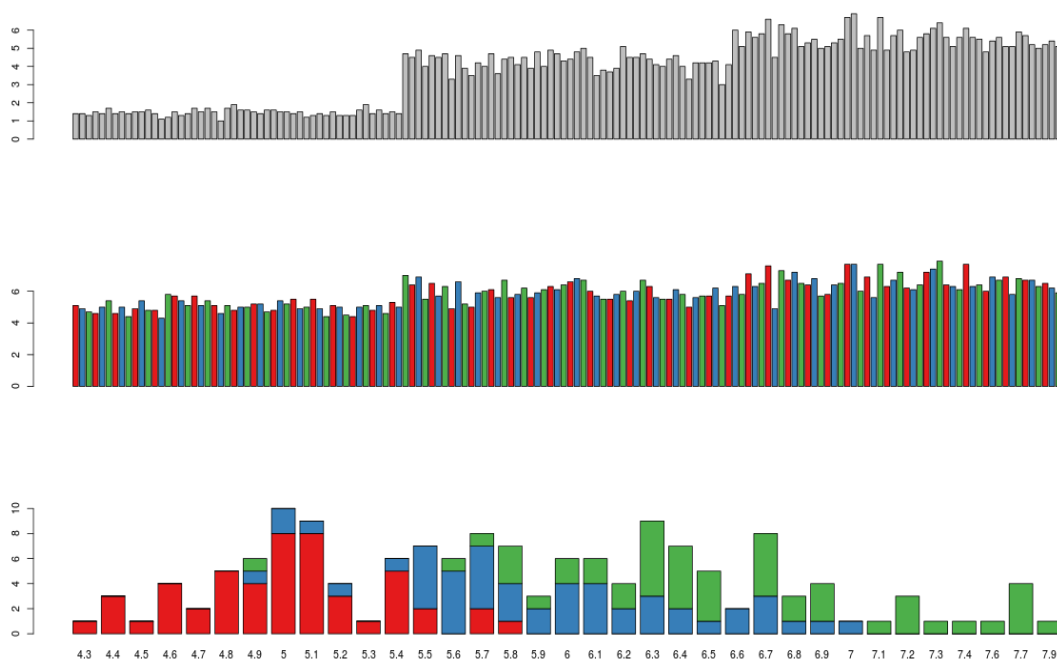
**fig 4.2 Line Chart Example**

- **Bar Chart**

Bar Plots are suitable for showing comparison between cumulative totals across several groups. Stacked Plots are used for bar plots for various categories. Here's the code:

```
barplot(iris$Petal.Length) #Creating simple Bar Graph  
barplot(iris$Sepal.Length,col = brewer.pal(3,"Set1"))
```

```
barplot(table(iris$Species,iris$Sepal.Length),col = brewer.pal(3,"Set1")) #Stacked
```



**Fig 4.3 Stacked Bar Graph Example**

- **Box Plot (including group-by option )**

Box Plot shows 5 statistically significant numbers- the minimum, the 25th percentile, the median, the 75th percentile and the maximum. It is thus useful for visualizing the spread of the data and deriving inferences accordingly. Here's the basic code:

```
boxplot(iris$Petal.Length~iris$Species) #Creating Box Plot between two variable
```

Let's understand the code below:

In the example below, I have made 4 graphs on one screen. By using the ~ sign, I can visualize how the spread (of Sepal Length) is across various categories (of Species). In the last two graphs I have shown the example of color palettes. A color palette is a group of colors that is used to make the graph more appealing and help create visual distinctions in the data.

```
data(iris)
par(mfrow=c(2,2))
boxplot(iris$Sepal.Length,col="red")
boxplot(iris$Sepal.Length~iris$Species,col="red")
```



```
oxplot(iris$Sepal.Length~iris$Species,col=heat.colors(3))
```

```
boxplot(iris$Sepal.Length~iris$Species,col=topo.colors(3))
```

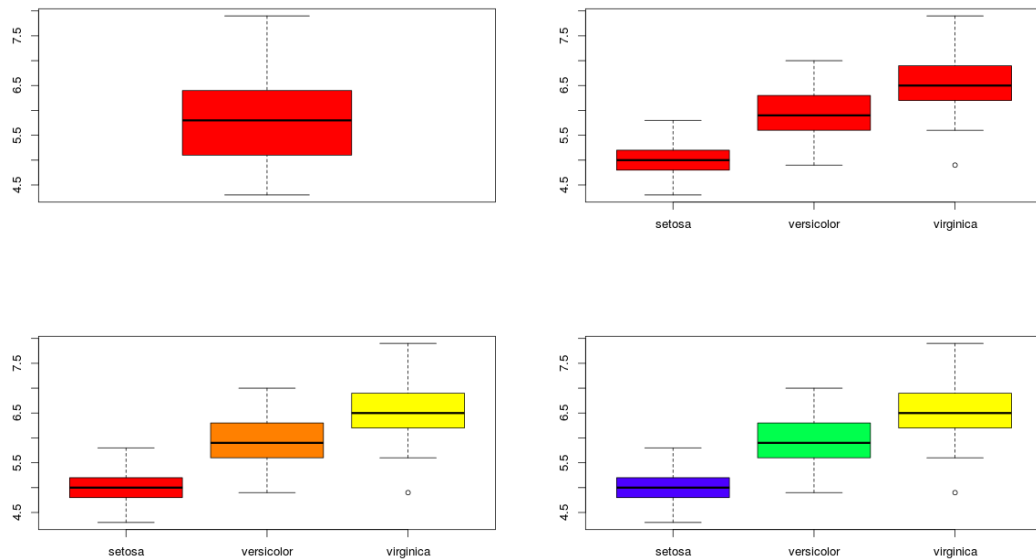


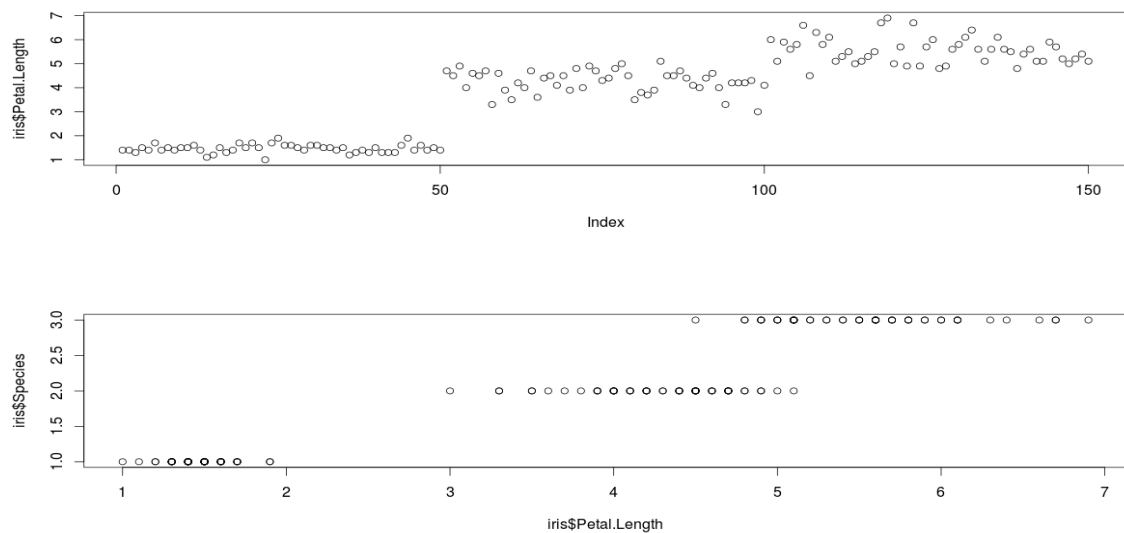
fig 4.4 Box Plot Example

- **Scatter Plot (including 3D and other features)**

Scatter plots help in visualizing data easily and for simple data inspection. Here's the code for simple scatter and multivariate scatter plot:

```
plot(x=iris$Petal.Length) #Simple Scatter Plot
```

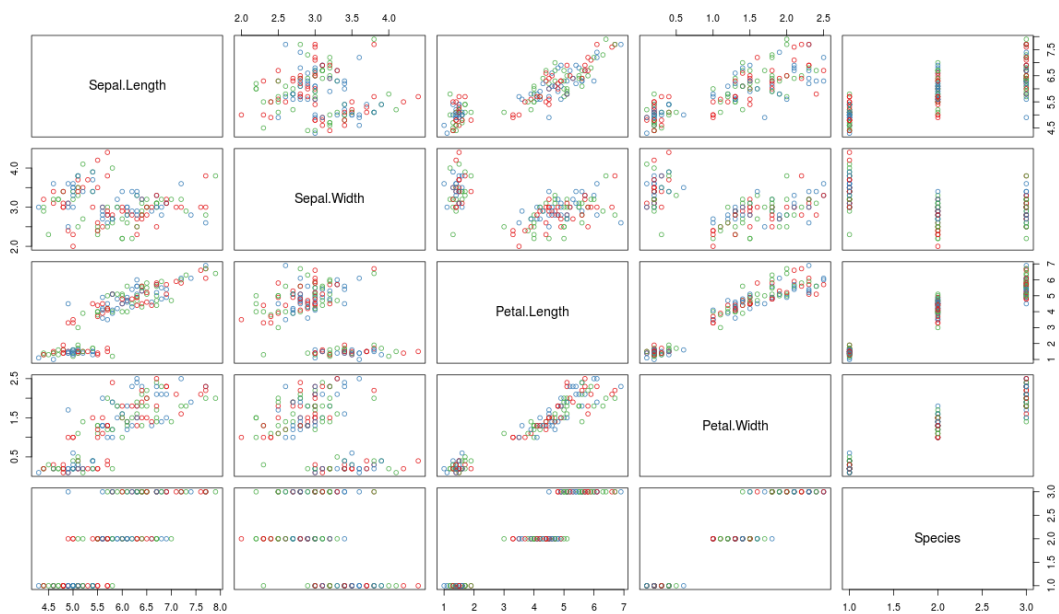
```
plot(x=iris$Petal.Length,y=iris$Species) #Multivariate Scatter Plot
```



**fig 4.5 Scatter plot Example**

Scatter Plot Matrix can help visualize multiple variables across each other.

```
plot(iris,col=brewer.pal(3,"Set1"))
```



**fig 4.6 3D Scatter plot Example**

You might be thinking that I have not put pie charts in the list of basic charts. That's intentional, not a miss out. This is because data visualization professionals' frown on the usage of pie charts to represent data. This is because the human eye cannot

visualize circular distances as accurately as linear distance. Simply put- anything that can be put in a pie chart is better represented as a line graph. However, if you like pie-chart, use:

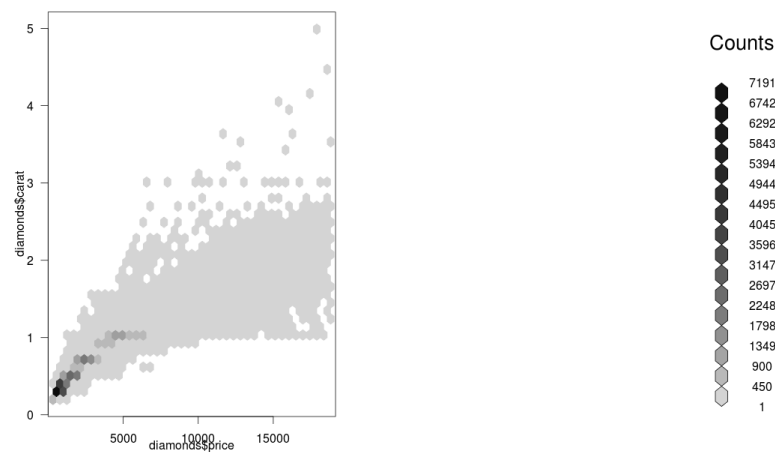
```
pie(table(iris$Species))
```

- **Hexagon Binning**

We can use the hexbin package in case we have multiple points in the same place (overplotting). Hexagon binning is a form of bivariate histogram useful for visualizing the structure in datasets with large n. Here's the code:

```
>library(hexbin)
>a=hexbin(diamonds$price,diamonds$carat,xbins=40)
>library(RColorBrewer)

>plot(a)
```

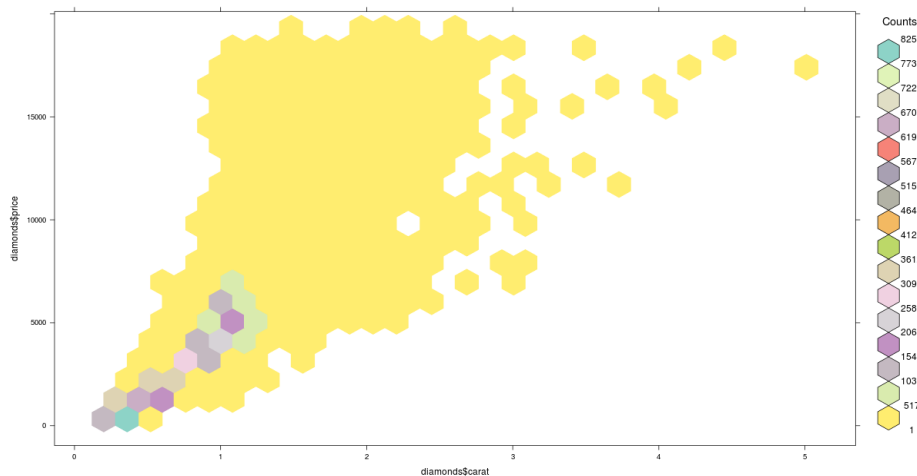


**fig 4.7 Hexagon Binding Example**

We can also create a color palette and then use the hexbin plot function for a better visual effect. Here's the code:

```
>library(RColorBrewer)
>rf <- colorRampPalette(rev(brewer.pal(40,'Set3')))
```

```
> hexbinplot(diamonds$price~diamonds$carat, data=diamonds, colramp=rf)
```



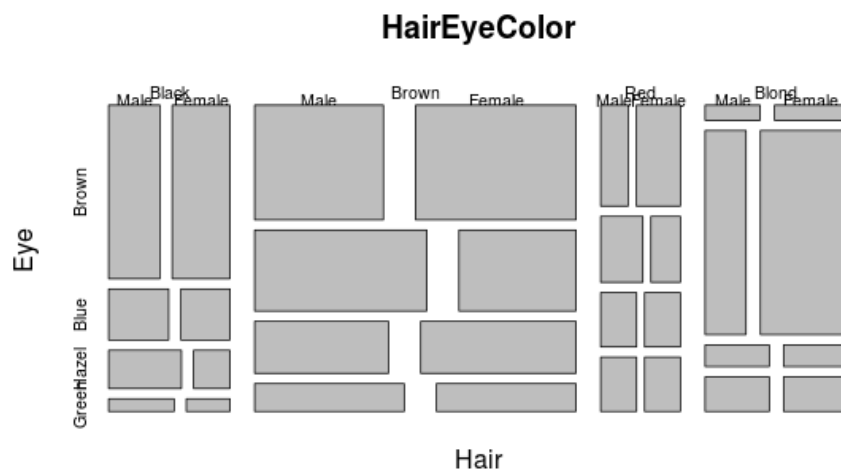
**Fig 4.8 Coloured Hexagon Binding Plot Example**

- **Mosaic Plot**

A mosaic plot can be used for plotting categorical data very effectively with the area of the data showing the relative proportions.

```
> data(HairEyeColor)
```

```
> mosaicplot(HairEyeColor)
```



**fig 4.9 Mosaic plot Example**

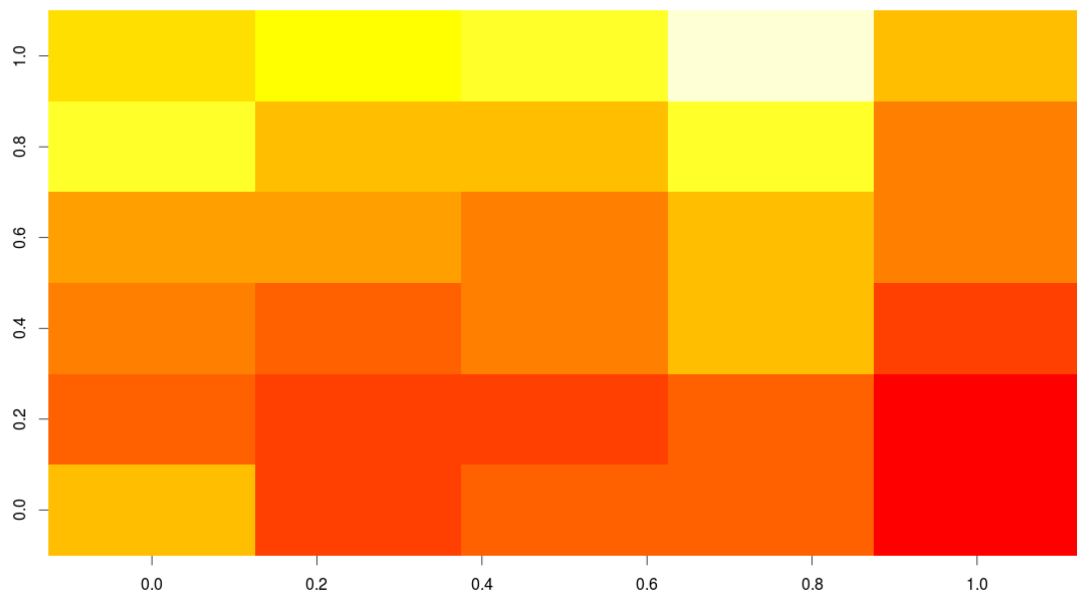
- **Heat Map**

Heat maps enable you to do exploratory data analysis with two dimensions as the axis and the third dimension shown by intensity of color. However you need to convert the dataset to a matrix format. For example:

```
> heatmap(as.matrix(mtcars))
```

You can use `image()` command also for this type of visualization as:

```
> image(as.matrix(b[2:7]))
```



**fig 4.10 Heat Map Example**

- **Map Visualization**

The latest thing in R is data visualization through javascript libraries. Leaflet is one of the most popular open-source javascript libraries for interactive maps. It is based at <https://rstudio.github.io/leaflet/>

```
library(magrittr)  
library(leaflet)
```

```
m <- leaflet() %>%
addTiles() %>% # Add default OpenStreetMap map tiles
addMarkers(lng=77.2310, lat=28.6560, popup="The delicious food of chandni
chowk")

m # Print the map
```

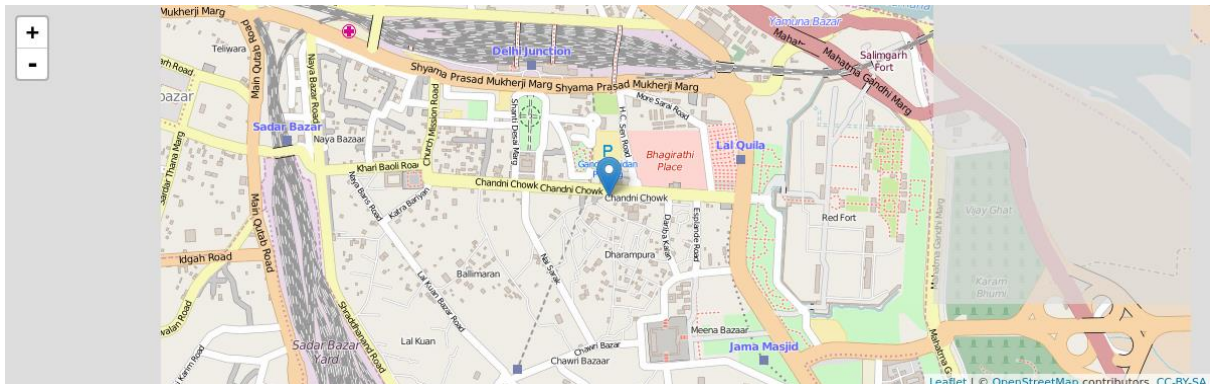


fig 4.11 Map Visualization Example

### • 3D Graphs

One of the easiest ways of impressing someone by R's capabilities is by creating a 3D graph in R without writing any line of code and within 3 minutes.

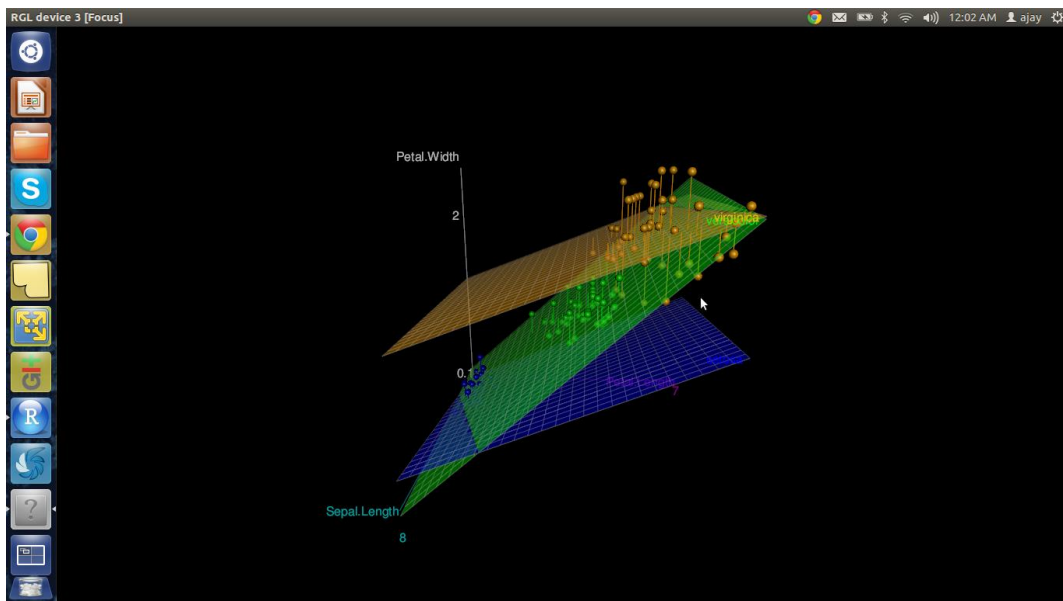
We use the package R Commander which acts as Graphical User Interface (GUI). Here are the steps:

- Simply install the Rcmdr package
- Use the 3D plot option from within graphs

The code below is not typed by the user but automatically generated.

Note: When we interchange the graph axes, you should see graphs with the respective code how we pass axis labels using xlab, ylab, and the graph title using Main and color using the col parameter.

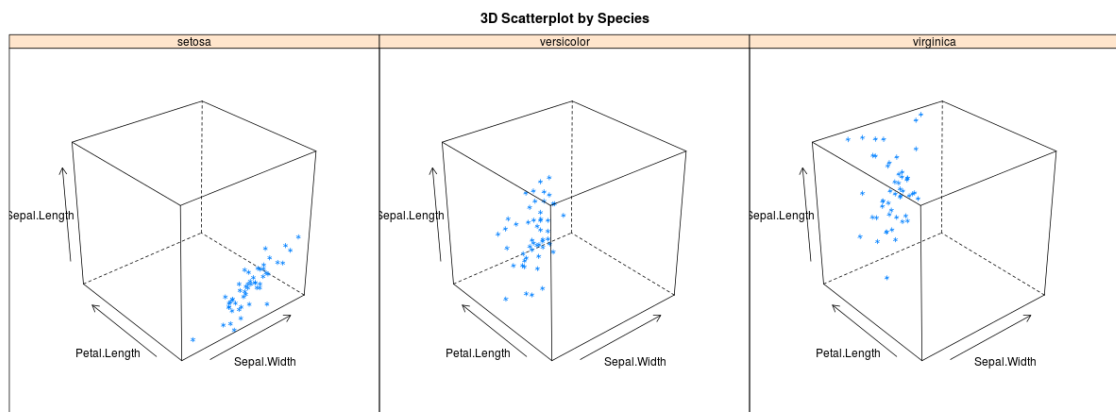
```
>data(iris, package="datasets")
>scatter3d(Petal.Width~Petal.Length+Sepal.Length|Species, data=iris, fit="linear"
>residuals=TRUE, parallel=FALSE, bg="black", axis.scales=TRUE, grid=TRUE,
ellipsoid=FALSE)
```



**fig 4.12 3D graphs**

You can also make 3D graphs using the Lattice package . Lattice can also be used for xyplots. Here's the code:

```
>attach(iris)# 3d scatterplot by factor level
>cloud(Sepal.Length~Sepal.Width*Petal.Length|Species, main="3D Scatterplot by Species")
>xyplot(Sepal.Width ~ Sepal.Length, iris, groups = iris$Species, pch= 20)
```



**fig 4.13 3D graphs**

- **Correlogram (GUIs)**

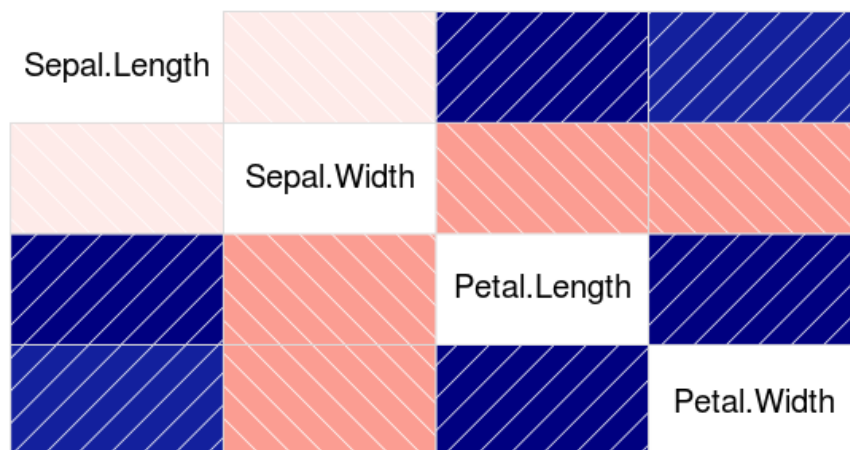
Correlograms help us visualize the data in correlation matrices. There are three principal GUI packages in R. RCommander with KMggplots, Rattle for data mining and Deducer for Data Visualization. These help to automate many tasks.

For example:

```
> cor(iris[1:4])
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
Sepal.Length	1.0000000	-0.1175698	0.8717538	0.8179411
Sepal.Width	-0.1175698	1.0000000	-0.4284401	-0.3661259
Petal.Length	0.8717538	-0.4284401	1.0000000	0.9628654
Petal.Width	0.8179411	-0.3661259	0.9628654	1.0000000

```
> corrgram(iris)
```



**fig 4.14 Correlogram (GUIs) Example**





## 5. Uber Data Analysis

### Importing the Essential Packages

In the first step of our R project, we will import the essential packages that we will use in this uber data analysis project. Some of the important libraries of R that we will use are –

- `ggplot2`

This is the backbone of this project. `ggplot2` is the most popular data visualization library that is most widely used for creating aesthetic visualization plots.

- `ggthemes`

This is more of an add-on to our main `ggplot2` library. With this, we can better create extra themes and scales with the mainstream `ggplot2` package.

- `lubridate`

Our dataset involves various timeframes. In order to understand our data in separate time categories, we will make use of the `lubridate` package.

- `dplyr`

This package is the lingua franca of data manipulation in R.

- `tidyr`

This package will help you to tidy your data. The basic principle of `tidyr` is to tidy the columns where each variable is present in a column, each observation is represented by a row and each value depicts a cell.

- `DT`

With the help of this package, we will be able to interface with the JavaScript Library called – `Datatables`.

- `scales`

With the help of graphical scales, we can automatically map the data to the correct scales with well-placed axes and legends.

### **Creating vector of colors to be implemented in our plots**

In this step of the R project, we will create a vector of our colors that will be included in our plotting functions. You can also select your own set of colors.

Code:

```
colors = c("#CC1011", "#665555", "#05a399", "#cfcaca", "#f5e840", "#0683c9", "#e075b0")
```

### **Reading the Data into their designated variables**

Now, we will read several csv files that contain the data from April 2014 to September 2014. We will store these in corresponding data frames like `apr_data`, `may_data`, etc. After we have read the files, we will combine all of this data into a single dataframe called `'data_2014'`.

Then, in the next step, we will perform the appropriate formatting of the `Date.Time` column. Then, we will proceed to create factors of time objects like day, month, year etc.

Code:

```
apr_data <- read.csv("uber-raw-data-apr14.csv")
may_data <- read.csv("uber-raw-data-may14.csv")
jun_data <- read.csv("uber-raw-data-jun14.csv")
jul_data <- read.csv("uber-raw-data-jul14.csv")
aug_data <- read.csv("uber-raw-data-aug14.csv")
sep_data <- read.csv("uber-raw-data-sep14.csv")

data_2014 <- rbind(apr_data, may_data, jun_data, jul_data, aug_data, sep_data)

data_2014$Date.Time <- as.POSIXct(data_2014$Date.Time, format = "%m/%d/%Y %H:%M:%S")

data_2014$Time <- format(as.POSIXct(data_2014$Date.Time, format = "%m/%d/%Y %H:%M:%S"), format = "%H:%M:%S")

data_2014$Date.Time <- ymd_hms(data_2014$Date.Time)
```

```
data_2014$day <- factor(day(data_2014$Date.Time))
data_2014$month <- factor(month(data_2014$Date.Time, label = TRUE))
data_2014$year <- factor(year(data_2014$Date.Time))
```

```
data_2014$dayofweek <- factor(wday(data_2014$Date.Time, label = TRUE))
```

### **Plotting the trips by the hours in a day**

In the next step of our R project, we will use the ggplot function to plot the number of trips that the passengers had made in a day. We will also use dplyr to aggregate our data. In the resulting visualizations, we can understand how the number of passengers fares throughout the day. We observe that the number of trips are higher in the evening around 5:00 and 6:00 PM.

```
hour_data <- data_2014 %>%
  group_by(hour) %>%
  dplyr::summarize(Total = n())
datatable(hour_data)
```

### **Plotting data by trips during every day of the month**

In this section of the R project, we will learn how to plot our data based on every day of the month. We observe from the resulting visualization that the 30th of the month had the highest trips in the year which is mostly contributed by the month of April.

Code:

```
day_group <- data_2014 %>%
  group_by(day) %>%
  dplyr::summarize(Total = n())
datatable(day_group)
```

### **Number of Trips taking place during months in a year**

In this section, we will visualize the number of trips that are taking place each month of the year. In the output visualization, we observe that most trips were made during the month of September. Furthermore, we also obtain visual reports of the number of trips that were made on every day of the week.

Code:

```
month_group <- data_2014 %>%
  group_by(month) %>%
```

```
dplyr::summarize(Total = n())
```

```
datatable(month_group)
```

### **Finding out the number of Trips by bases**

With the following code, we plot the number of trips that have been taken by the passengers from each of the bases. There are five bases in all out of which, we observe that B02617 had the highest number of trips. Furthermore, this base had the highest number of trips in the month B02617. Thursday observed the highest trips in the three bases – B02598, B02617, B02682.

Code:

```
ggplot(data_2014, aes(Base)) +  
  geom_bar(fill = "dark red") +  
  scale_y_continuous(labels = comma) +  
  
  ggtitle("Trips by Bases")
```

### **Creating a Heatmap visualization of day, hour and month**

Then, we will plot heatmaps using ggplot(). We will plot five heatmap plots –

- First, we will plot Heatmap by Hour and Day.
- Second, we will plot Heatmap by Month and Day.
- Third, a Heatmap by Month and Day of the Week.
- Fourth, a Heatmap that delineates Month and Bases.
- Finally, we will plot the heatmap, by bases and day of the week.

Code:

```
day_and_hour <- data_2014 %>%  
  group_by(day, hour) %>%  
  dplyr::summarize(Total = n())
```

```
datatable(day_and_hour)
```

## Creating a map visualization of rides in New York

In the final section, we will visualize the rides in New York city by creating a geo-plot that will help us to visualize the rides during 2014 (Apr – Sep) and by the bases in the same period.

Code:

```
min_lat <- 40.5774
max_lat <- 40.9176
min_long <- -74.15
max_long <- -73.7004

ggplot(data_2014, aes(x=Lon, y=Lat)) +
  geom_point(size=1, color = "blue") +
  scale_x_continuous(limits=c(min_long, max_long)) +
  scale_y_continuous(limits=c(min_lat, max_lat)) +
  theme_map() +
  ggtitle("NYC MAP BASED ON UBER RIDES DURING 2014 (APR-SEP)")

ggplot(data_2014, aes(x=Lon, y=Lat, color = Base)) +
  geom_point(size=1) +
  scale_x_continuous(limits=c(min_long, max_long)) +
  scale_y_continuous(limits=c(min_lat, max_lat)) +
  theme_map() +
  ggtitle("NYC MAP BASED ON UBER RIDES DURING 2014 (APR-SEP) by
BASE")
```



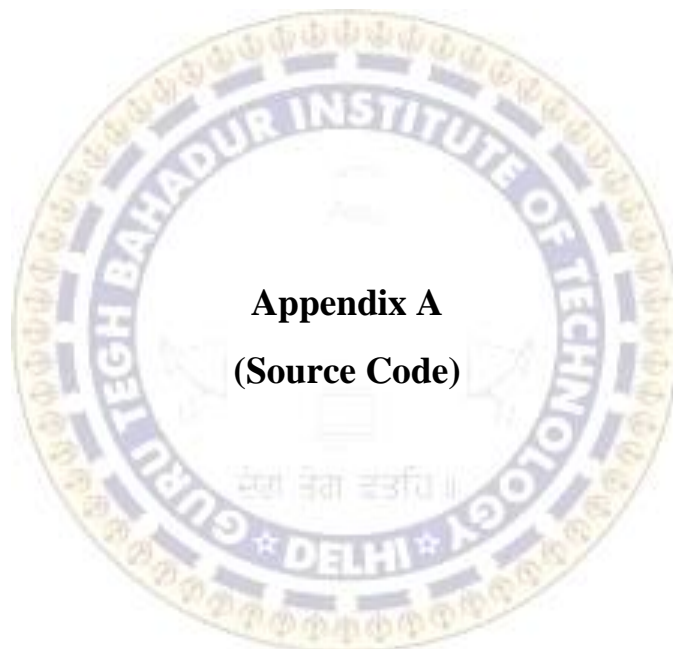
## **Summary and Conclusion**

## Summary and Conclusion

At the end of the Uber data analysis R project, we observed how to create data visualizations. We made use of packages like ggplot2 that allowed us to plot various types of visualizations that pertained to several time-frames of the year. With this, we could conclude how time affected customer trips. Finally, we made a geo plot of New York that provided us with the details of how various users made trips from different bases.







## **Appendix A**

### **(Source Code)**

## Code

```
#data analysis using uber dataset
#source : UBER 2014 Sample dataset

#Importing libraries
library(ggplot2)
library(ggthemes)
library(lubridate)
library(dplyr)
library(tidyr)
library(DT)
library(scales)

#creating vector of colors to be implemented in code
colors = c("#CC1011", "#665555", "#05a399", "#cfcaca", "#f5e840", "#0683c9",
"#e075b0")

#reading csv files
apr_data <- read.csv("C:/Users/verma/Downloads/Uber-dataset/uber-raw-data-
apr14.csv/uber-raw-data-apr14.csv")
may_data <- read.csv("C:/Users/verma/Downloads/Uber-dataset/uber-raw-data-
may14.csv/uber-raw-data-may14.csv")
jun_data <- read.csv("C:/Users/verma/Downloads/Uber-dataset/uber-raw-data-
jun14.csv/uber-raw-data-jun14.csv")
jul_data <- read.csv("C:/Users/verma/Downloads/Uber-dataset/uber-raw-data-
jul14.csv/uber-raw-data-jul14.csv")
aug_data <- read.csv("C:/Users/verma/Downloads/Uber-dataset/uber-raw-data-
aug14.csv/uber-raw-data-aug14.csv")
sep_data <- read.csv("C:/Users/verma/Downloads/Uber-dataset/uber-raw-data-
sep14.csv/uber-raw-data-sep14.csv")

#combining all the data
data_2014 <- rbind(apr_data,may_data, jun_data, jul_data, aug_data, sep_data)

#Formatting the dataset
data_2014$Date.Time <- as.POSIXct(data_2014$Date.Time, format = "%m/%d/%Y
%H:%M:%S")

data_2014$Time <- format(as.POSIXct(data_2014$Date.Time, format =
"%m/%d/%Y %H:%M:%S"), format="%H:%M:%S")

data_2014$Date.Time <- ymd_hms(data_2014$Date.Time)

data_2014$day <- factor(day(data_2014$Date.Time))
data_2014$month <- factor(month(data_2014$Date.Time, label = TRUE))
```

```

data_2014$year <- factor(year(data_2014$Date.Time))
data_2014$dayofweek <- factor(wday(data_2014$Date.Time, label = TRUE))

data_2014$hour <- factor(hour(hms(data_2014$Time)))
data_2014$minute <- factor(minute(hms(data_2014$Time)))
data_2014$second <- factor(second(hms(data_2014$Time)))

#Plotting trips by hour in a day
hour_data <- data_2014 %>%
  group_by(hour) %>%
  dplyr::summarize(Total = n())
datatable(hour_data)

#Visualizing the trip data by hours
ggplot(hour_data, aes(hour, Total)) +
  geom_bar( stat = "identity", fill = "steelblue", color = "red") +
  ggtitle("Trips Every Hour") +
  theme(legend.position = "none") +
  scale_y_continuous(labels = comma)

#Visualizing the trip data by hours and month
month_hour <- data_2014 %>%
  group_by(month, hour) %>%
  dplyr::summarize(Total = n())

ggplot(month_hour, aes(hour, Total, fill = month)) +
  geom_bar( stat = "identity") +
  ggtitle("Trips by Hour and Month") +
  scale_y_continuous(labels = comma)

#Plotting data by trips during every day of the month
day_group <- data_2014 %>%
  group_by(day) %>%
  dplyr::summarize(Total = n())
datatable(day_group)

#Visualizing data by trips during every day of the month
ggplot(day_group, aes(day, Total)) +
  geom_bar( stat = "identity", fill = "steelblue") +
  ggtitle("Trips Every Day") +
  theme(legend.position = "none") +
  scale_y_continuous(labels = comma)

#Visualizing trips by day and month
day_month_group <- data_2014 %>%
  group_by(month, dayofweek) %>%
  dplyr::summarize(Total = n())

ggplot(day_month_group, aes(month, Total, fill = dayofweek)) +
  geom_col( stat = "identity" , position = "dodge") +

```

```

ggtitle("Trips by Day and Month") +
scale_y_continuous(labels = comma) +
scale_fill_manual(values = colors)

#Visualizing Number of Trips taking place during months in a year
month_group <- data_2014 %>%
  group_by(month) %>%
  dplyr::summarize(Total = n())
datatable(month_group)

ggplot(month_group , aes(month, Total, fill = month)) +
  geom_bar( stat = "identity") +
  ggtitle("Trips by Month") +
  theme(legend.position = "none") +
  scale_y_continuous(labels = comma) +
  scale_fill_manual(values = colors)

#Visualizing trips by day and month
month_weekday <- data_2014 %>%
  group_by(month, dayofweek) %>%
  dplyr::summarize(Total = n())

ggplot(month_weekday, aes(month, Total, fill = dayofweek)) +
  geom_bar( stat = "identity", position = "dodge") +
  ggtitle("Trips by Day and Month") +
  scale_y_continuous(labels = comma) +
  scale_fill_manual(values = colors)

#Finding out the number of Trips by bases
ggplot(data_2014, aes(Base)) +
  geom_bar(fill = "darkred") +
  scale_y_continuous(labels = comma) +
  ggtitle("Trips by Bases")

#trips by bases and month
ggplot(data_2014, aes(Base, fill = month)) +
  geom_bar(position = "dodge") +
  scale_y_continuous(labels = comma) +
  ggtitle("Trips by Bases and Month") +
  scale_fill_manual(values = colors)

#trips by bases and day of week
ggplot(data_2014, aes(Base, fill = dayofweek)) +
  geom_bar(position = "dodge") +
  scale_y_continuous(labels = comma) +
  ggtitle("Trips by Bases and DayofWeek") +
  scale_fill_manual(values = colors)

#Creating a Heatmap visualization of day, hour and month

```

```

#Heatmap by Hour and Day
day_and_hour <- data_2014 %>%
  group_by(day, hour) %>%
  dplyr::summarize(Total = n())
datatable(day_and_hour)

ggplot(day_and_hour, aes(day, hour, fill = Total)) +
  geom_tile(color = "white") +
  ggtitle("Heat Map by Hour and Day")

#Heatmap by Month and Day
day_month_group <- data_2014 %>%
  group_by( day, month) %>%
  dplyr::summarize(Total = n())
datatable(day_month_group)

ggplot(day_month_group, aes(day, month, fill = Total)) +
  geom_tile(color = "white") +
  ggtitle("Heat Map by Month and Day")

#Heatmap by Month and Dayofweek
month_weekday <- data_2014 %>%
  group_by( dayofweek, month) %>%
  dplyr::summarize(Total = n())
datatable(month_weekday)

ggplot(month_weekday, aes(dayofweek, month, fill = Total)) +
  geom_tile(color = "white") +
  ggtitle("Heat Map by Month and Day of Week")

#Heatmap by Month and bases
month_base <- data_2014 %>%
  group_by(Base, month) %>%
  dplyr::summarize(Total = n())

ggplot(month_base, aes(Base, month, fill = Total)) +
  geom_tile(color = "white") +
  ggtitle("Heat Map by Month and Bases")

#Heat Map by Bases and Day of Week
dayofweek_bases <- data_2014 %>%
  group_by(Base, dayofweek) %>%
  dplyr::summarize(Total = n())
datatable(dayofweek_bases)

ggplot(dayofweek_bases, aes(Base, dayofweek, fill = Total)) +
  geom_tile(color = "white") +
  ggtitle("Heat Map by Bases and Day of Week")

```

#Creating a map visualization of rides in New York

```
min_lat <- 40.5774  
max_lat <- 40.9176  
min_long <- -74.15  
max_long <- -73.7004
```

```
ggplot(data_2014, aes(x=Lon, y=Lat)) +  
  geom_point(size=1, color = "blue") +  
  scale_x_continuous(limits=c(min_long, max_long)) +  
  scale_y_continuous(limits=c(min_lat, max_lat)) +  
  theme_map() +  
  ggtitle("NYC MAP BASED ON UBER RIDES DURING 2014 (APR-SEP)")
```

```
ggplot(data_2014, aes(x=Lon, y=Lat, color = Base)) +  
  geom_point(size=1) +  
  scale_x_continuous(limits=c(min_long, max_long)) +  
  scale_y_continuous(limits=c(min_lat, max_lat)) +  
  theme_map() +  
  ggtitle("NYC MAP BASED ON UBER RIDES DURING 2014 (APR-SEP) by  
BASE")
```

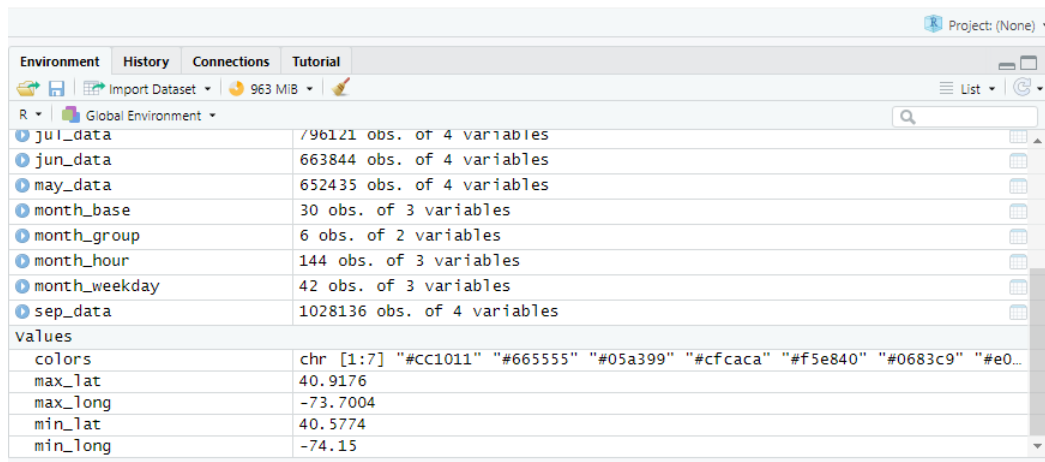




## **Appendix B**

### **(Screenshots)**

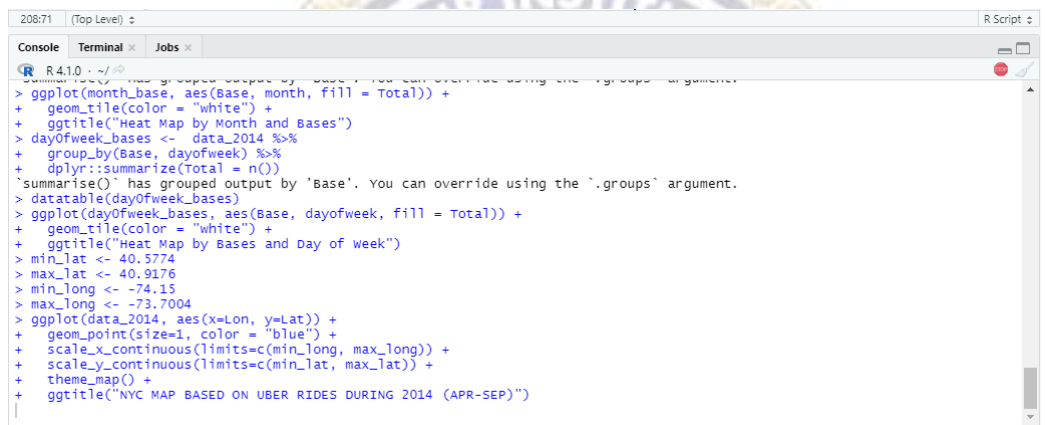
## Rstudio variables and data sources tab



The screenshot shows the RStudio Environment pane with the following variables and their values:

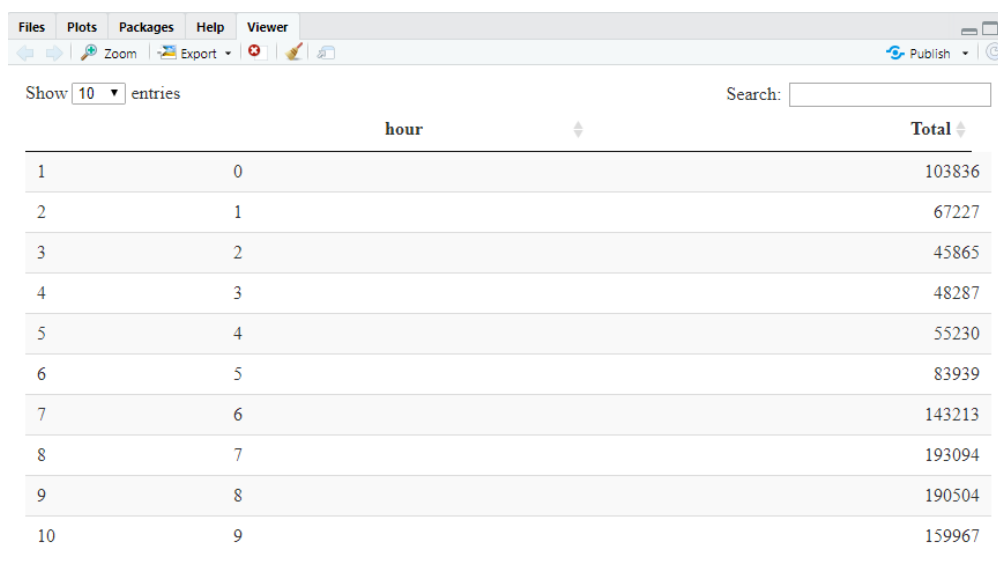
Variable	Value
jul_data	796121 obs. of 4 variables
jun_data	663844 obs. of 4 variables
may_data	652435 obs. of 4 variables
month_base	30 obs. of 3 variables
month_group	6 obs. of 2 variables
month_hour	144 obs. of 3 variables
month_weekday	42 obs. of 3 variables
sep_data	1028136 obs. of 4 variables
colors	chr [1:7] "#CC1011" "#665555" "#05a399" "#cfca" "#f5e840" "#0683c9" "#e0...
max_lat	40.9176
max_long	-73.7004
min_lat	40.5774
min_long	-74.15

## Rstudio output console



```
R 4.1.0 - ~/
> ggplot(month_base, aes(Base, month, fill = Total)) +
+   geom_tile(color = "white") +
+   ggtitle("Heat Map by Month and Bases")
> dayofweek_bases <- data_2014 %>%
+   group_by(Base, dayofweek) %>%
+   dplyr::summarize(Total = n())
'summarise()' has grouped output by 'Base'. You can override using the '.groups' argument.
> datatable(dayofweek_bases)
> ggplot(dayofweek_bases, aes(Base, dayofweek, fill = Total)) +
+   geom_tile(color = "white") +
+   ggtitle("Heat Map by Bases and Day of week")
> min_lat <- 40.5774
> max_lat <- 40.9176
> min_long <- -74.15
> max_long <- -73.7004
> ggplot(data_2014, aes(x=Lon, y=Lat)) +
+   geom_point(size=1, color = "blue") +
+   scale_x_continuous(limits=c(min_long, max_long)) +
+   scale_y_continuous(limits=c(min_lat, max_lat)) +
+   theme_map() +
+   ggtitle("NYC MAP BASED ON UBER RIDES DURING 2014 (APR-SEP)")
```

## Plotting the trips by the hours in a day

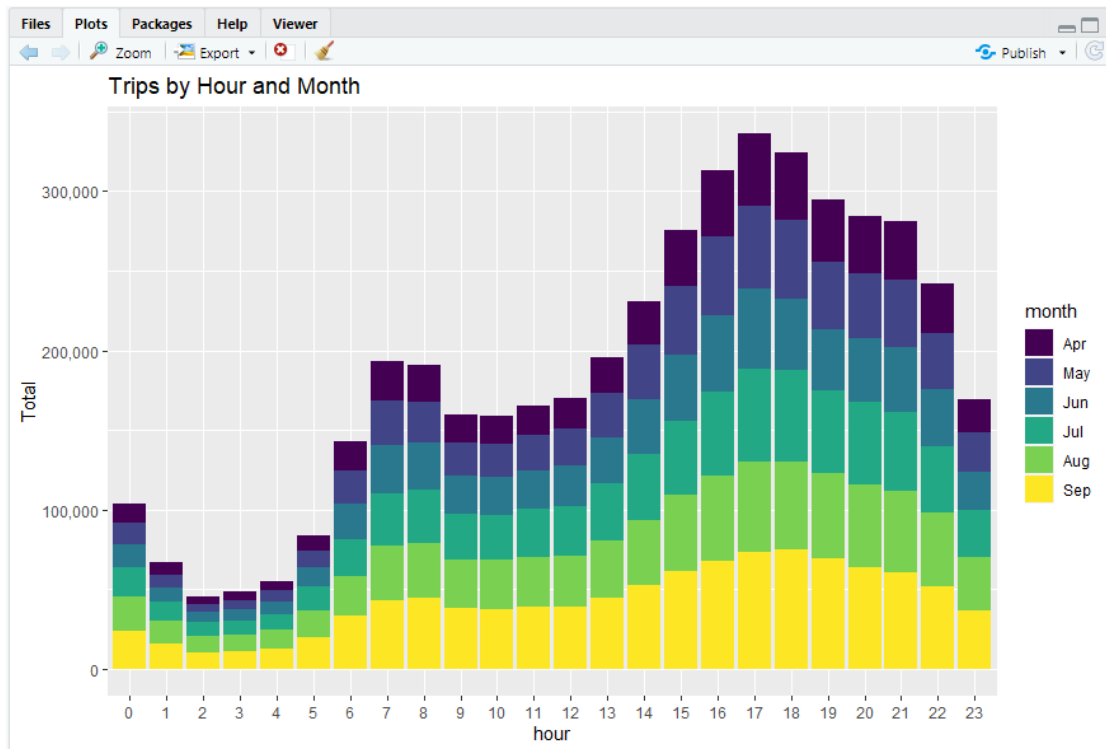


The screenshot shows the RStudio Plots pane with a table of trip data by hour. The table has 10 entries, showing the hour of the day and the total number of trips.

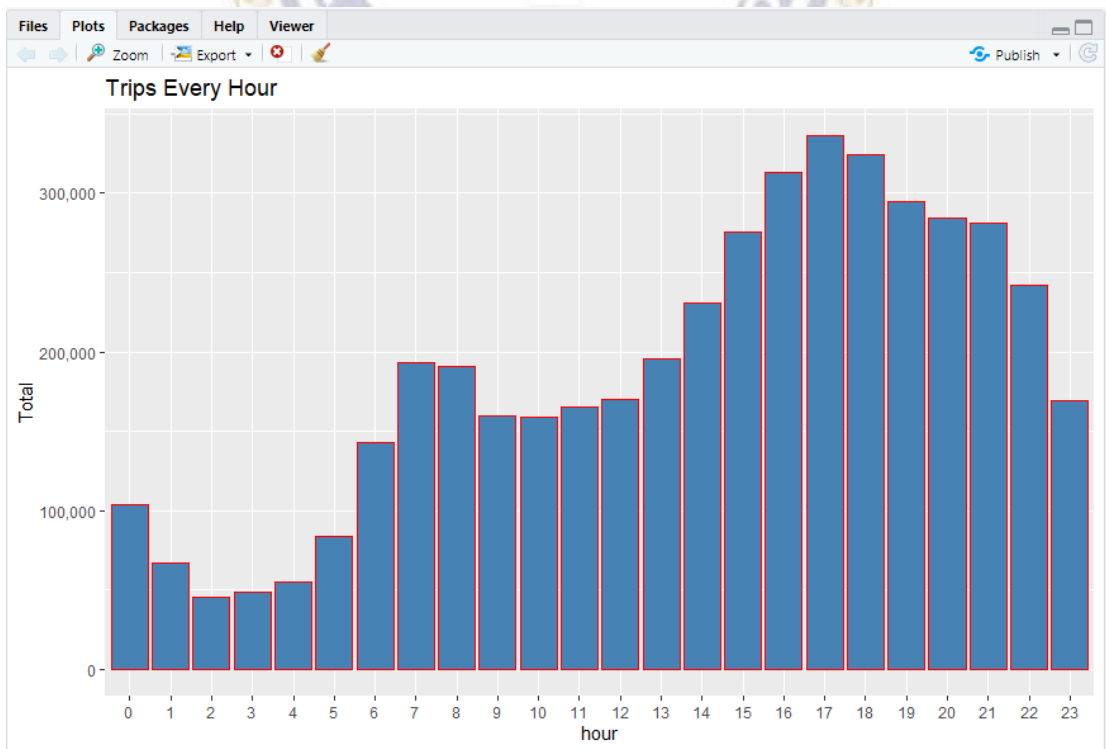
hour	Total
1	103836
2	67227
3	45865
4	48287
5	55230
6	83939
7	143213
8	193094
9	190504
10	159967



## Plotting the trips by the hours in a day



## Plotting the trips every hour



## Plotting the trips during every day of the month

Files

Plots

Packages

Help

Viewer

◀

▶

Zoom

Export

Publish

Show 

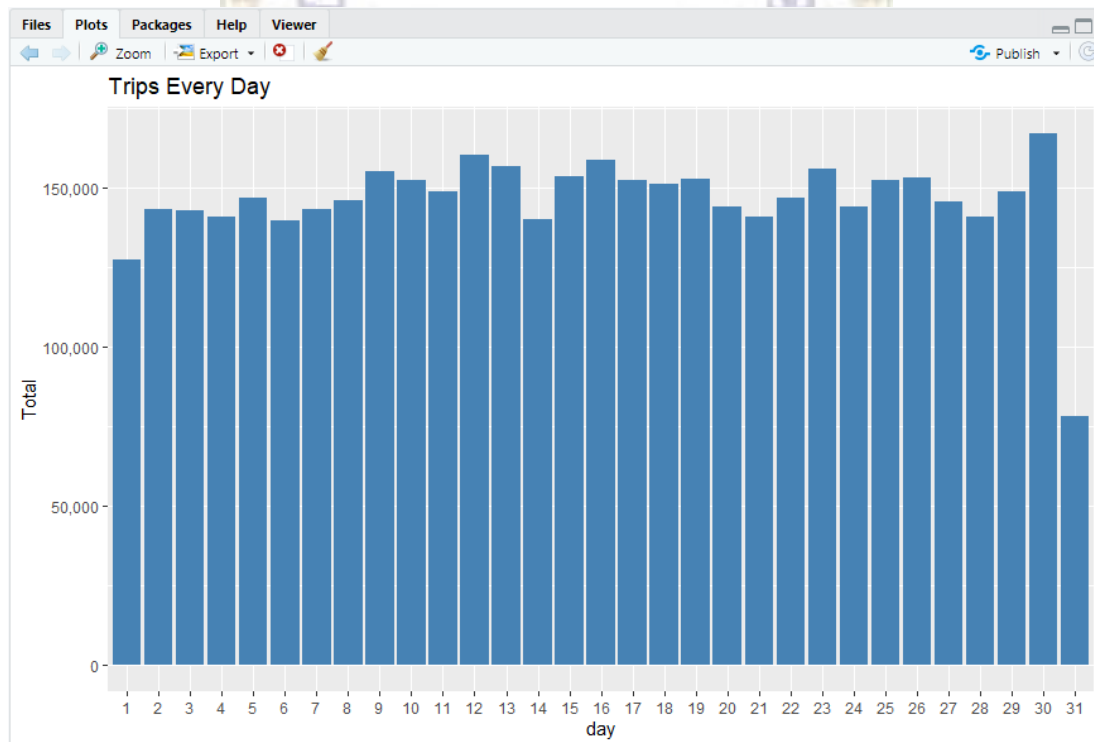
10

 entries

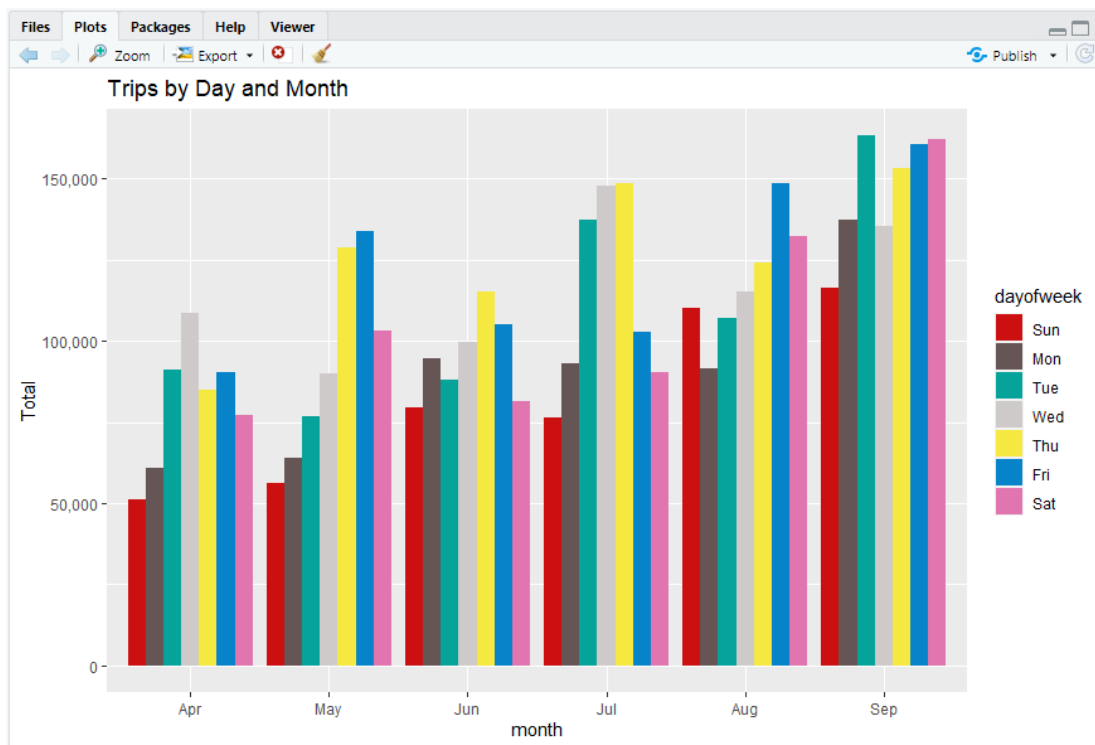
Search:

	day	Total
1	1	127430
2	2	143201
3	3	142983
4	4	140923
5	5	147054
6	6	139886
7	7	143503
8	8	145984
9	9	155135
10	10	152500

## Plotting the trips every day



## Plotting the trips by day and month



## Number of Trips taking place during months in a year

FilesPlotsPackagesHelpViewer

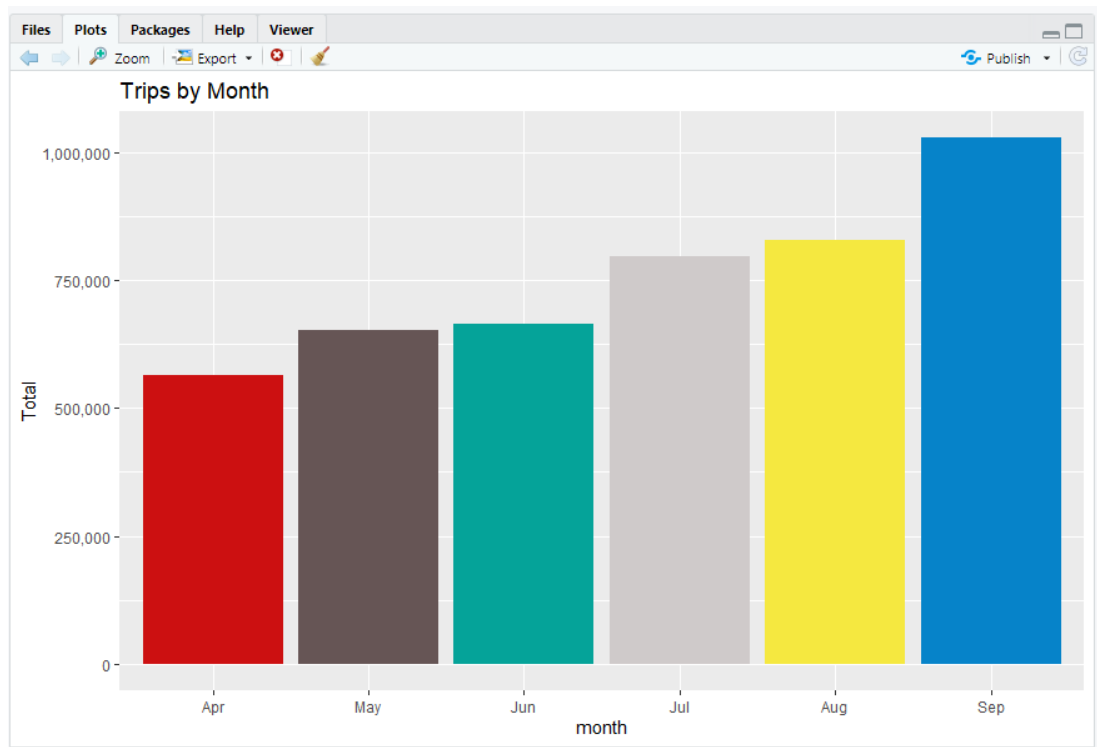
ZoomExportPublish

Show 10 entriesSearch:

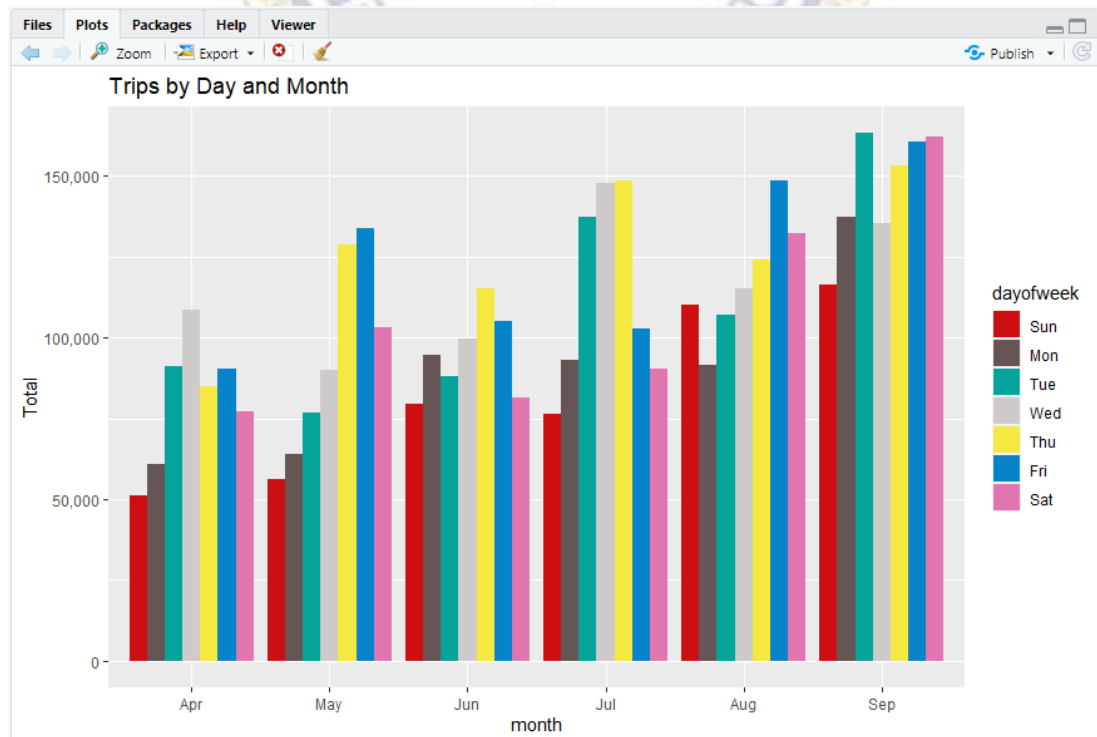
	month	Total
1	Apr	564516
2	May	652435
3	Jun	663844
4	Jul	796121
5	Aug	829275
6	Sep	1028136

Showing 1 to 6 of 6 entriesPrevious1Next

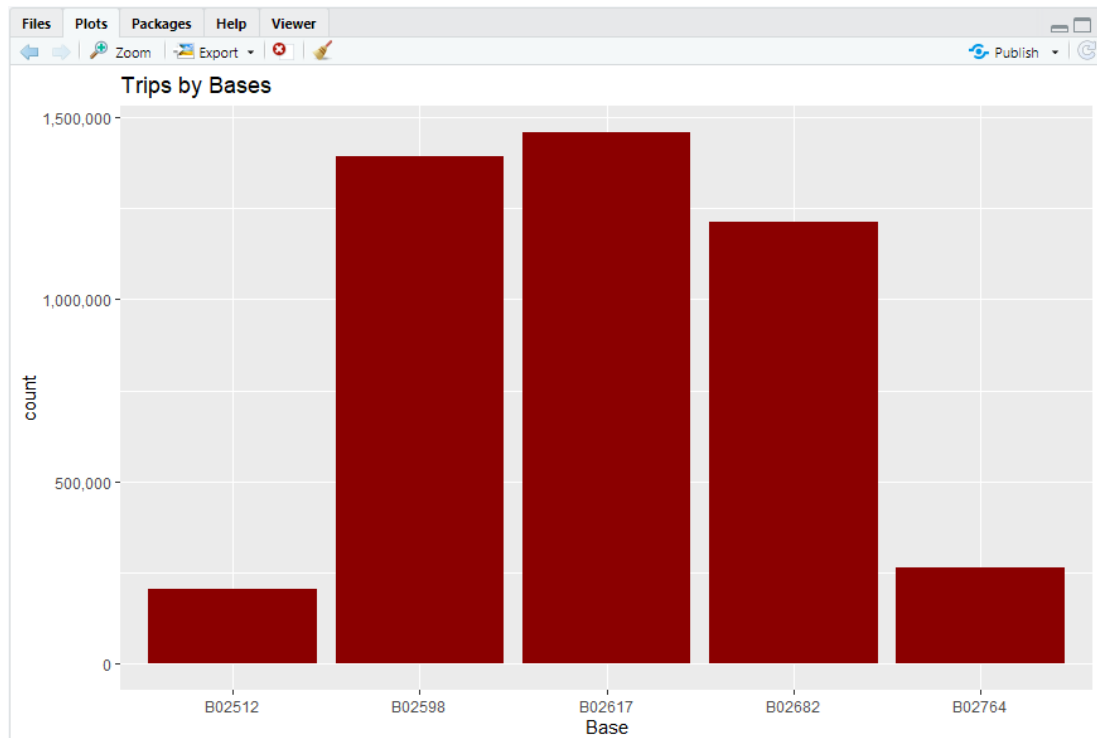
## Plotting trips by month



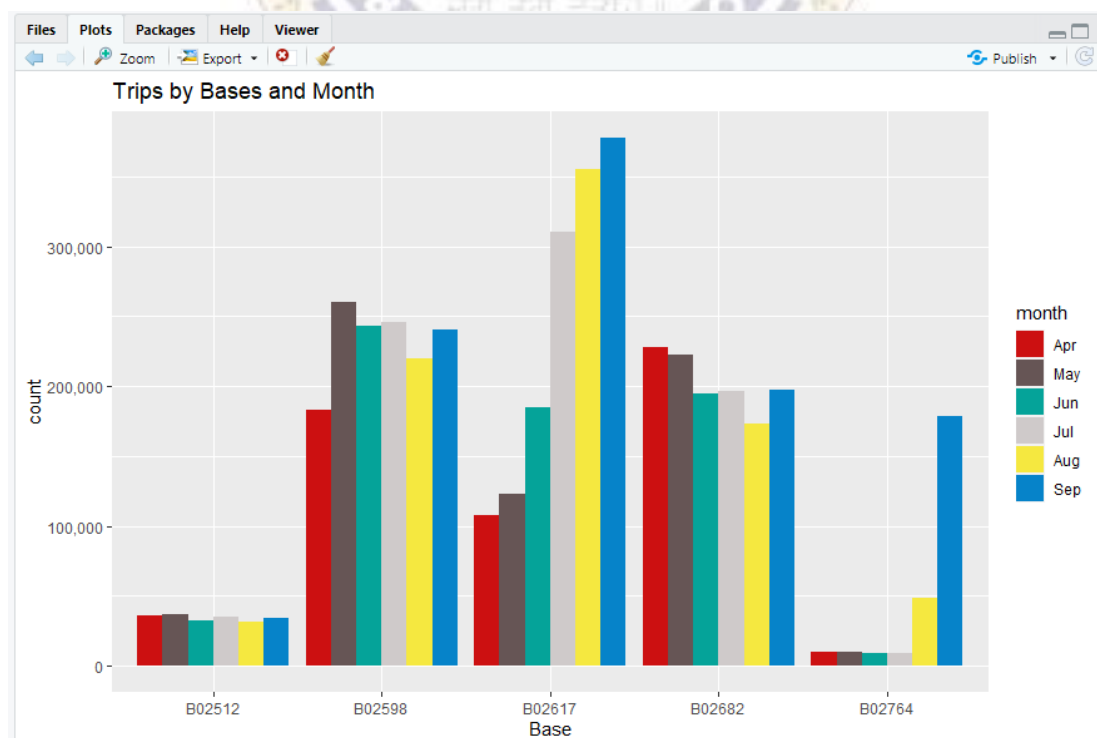
## Plotting trips by day and month



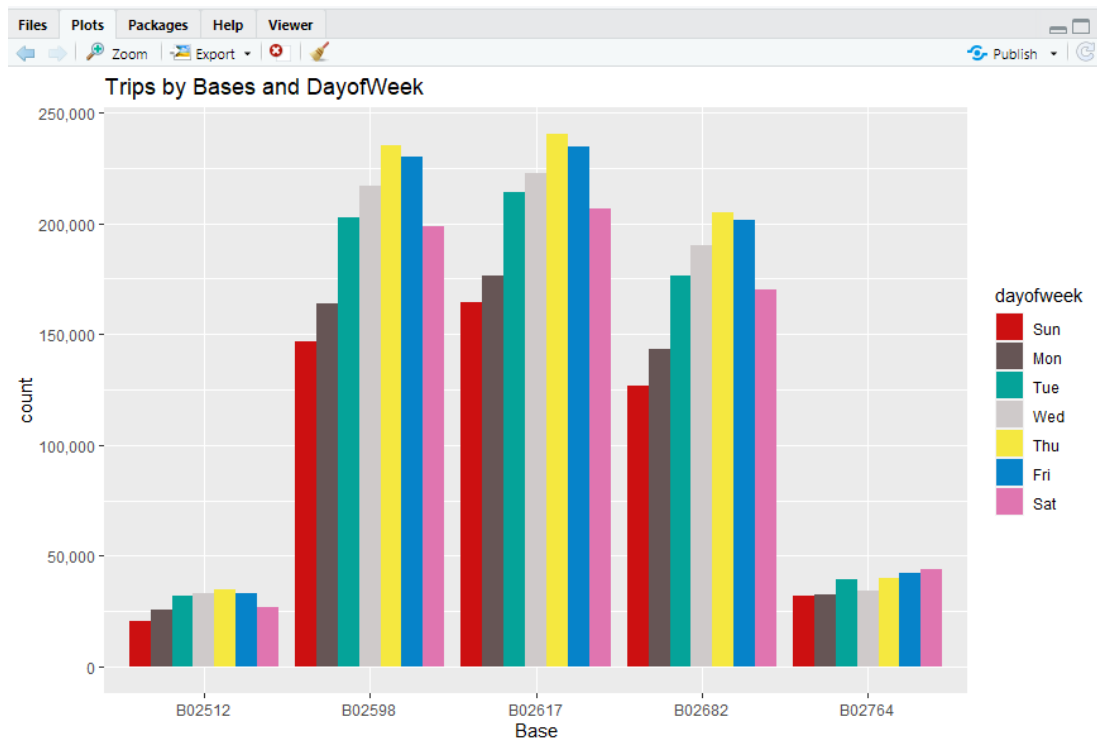
## Finding out the number of Trips by bases



## Plotting trips by bases and month



## Plotting trips by bases and days of week



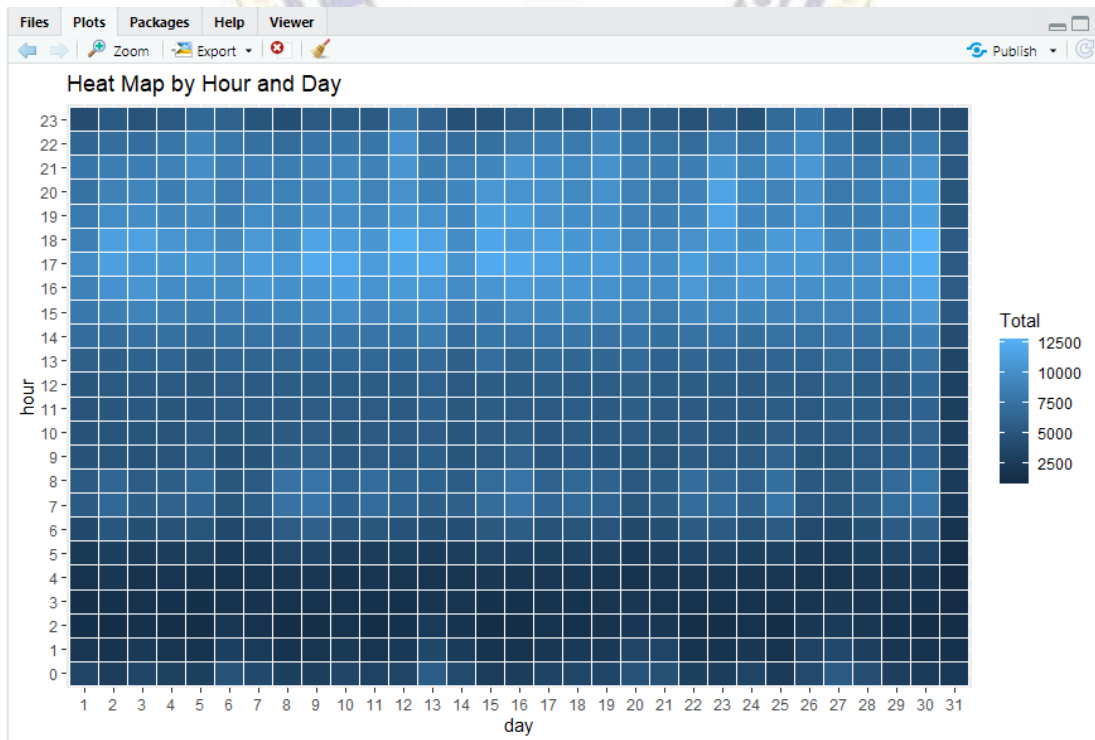
## Creating a Heatmap visualization of trips by hour and day

Files Plots Packages Help Viewer			
Zoom Export Publish			
Show 10 entries		Search:	
	day	hour	Total
1	1	0	3247
2	1	1	1982
3	1	2	1284
4	1	3	1331
5	1	4	1458
6	1	5	2171
7	1	6	3717
8	1	7	5470
9	1	8	5376
10	1	9	4688

Showing 1 to 10 of 744 entries

Previous 1 2 3 4 5 ... 75 Next

## Plotting the Heatmap of trips by hour and day



## Creating a Heatmap visualization of trips by month and day

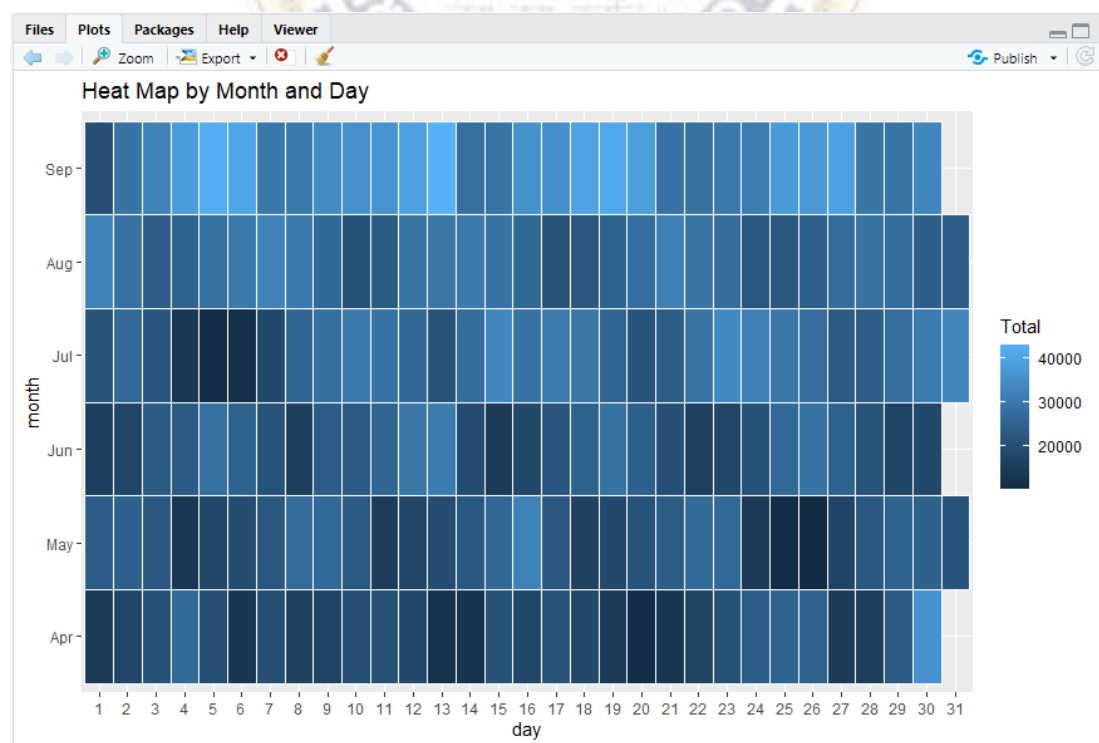
Files Plots Packages Help Viewer

Show 10 entries Search:

	day	month	Total
1	1	Apr	14546
2	1	May	23375
3	1	Jun	15967
4	1	Jul	21228
5	1	Aug	32353
6	1	Sep	19961
7	2	Apr	17474
8	2	May	24235
9	2	Jun	17503
10	2	Jul	26480

Showing 1 to 10 of 183 entries Previous 1 2 3 4 5 ... 19 Next

## Plotting the Heatmap of trips by month and day





## Creating a Heatmap visualization of trips by month and day of week

Files Plots Packages Help Viewer

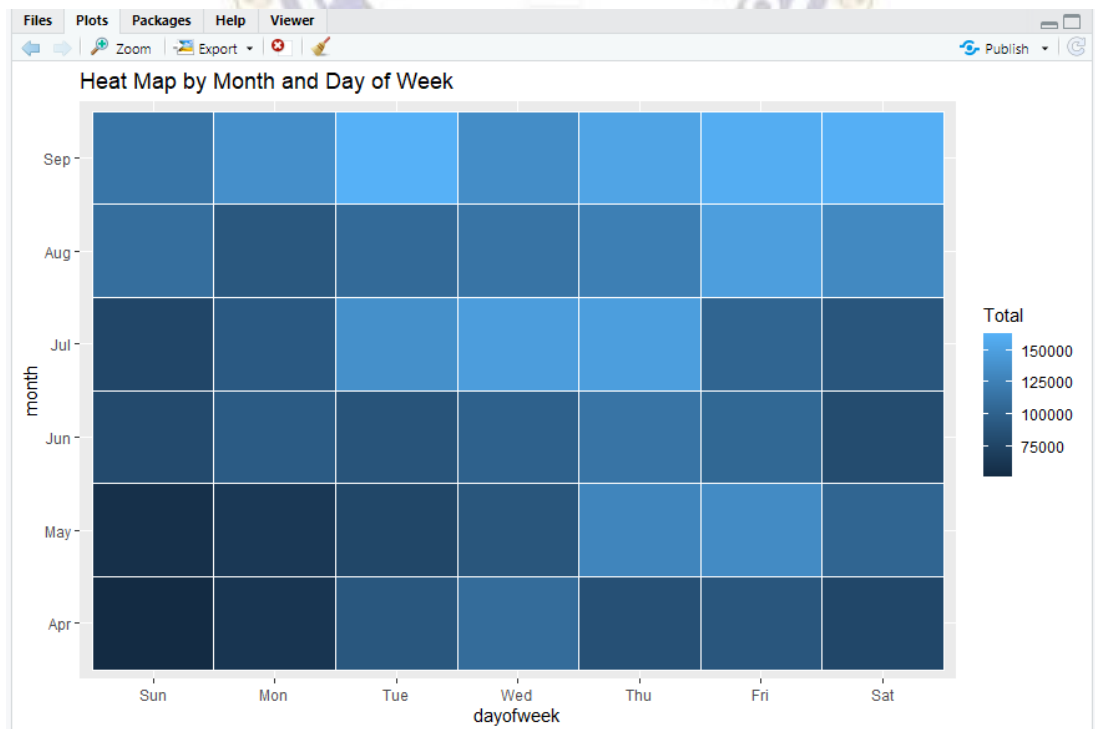
Zoom Export Publish

Show 10 entries Search:

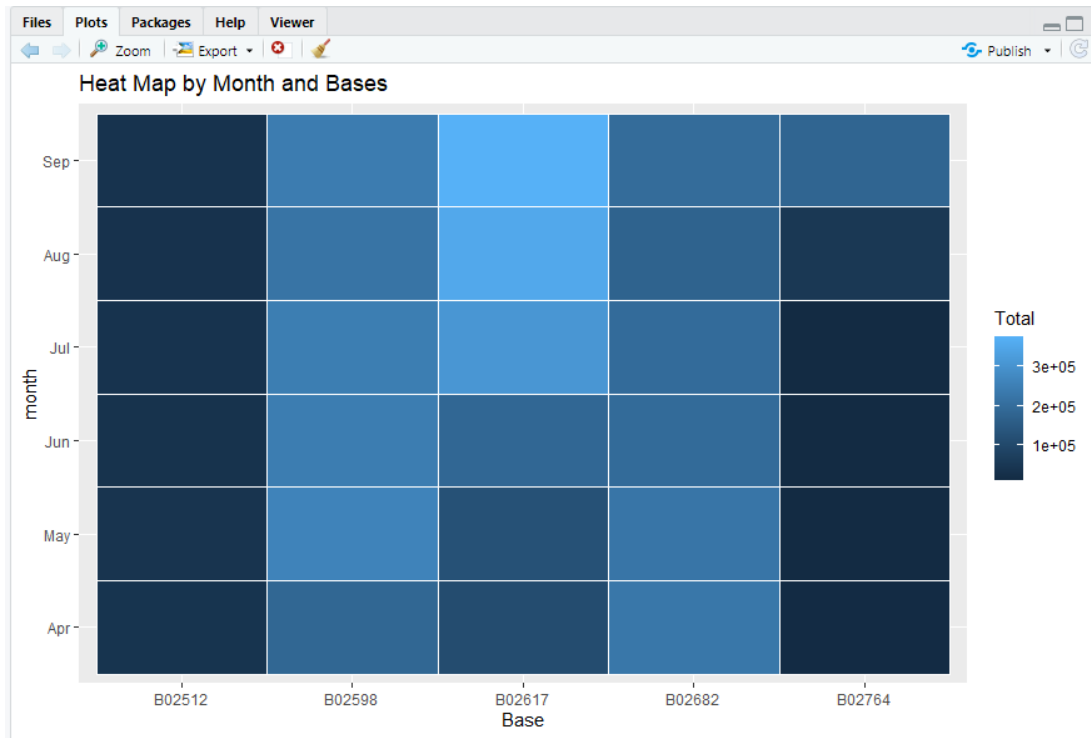
	dayofweek	month	Total
1	Sun	Apr	51251
2	Sun	May	56168
3	Sun	Jun	79656
4	Sun	Jul	76327
5	Sun	Aug	110246
6	Sun	Sep	116532
7	Mon	Apr	60861
8	Mon	May	63846
9	Mon	Jun	94655
10	Mon	Jul	93189

Showing 1 to 10 of 42 entries Previous 1 2 3 4 5 Next

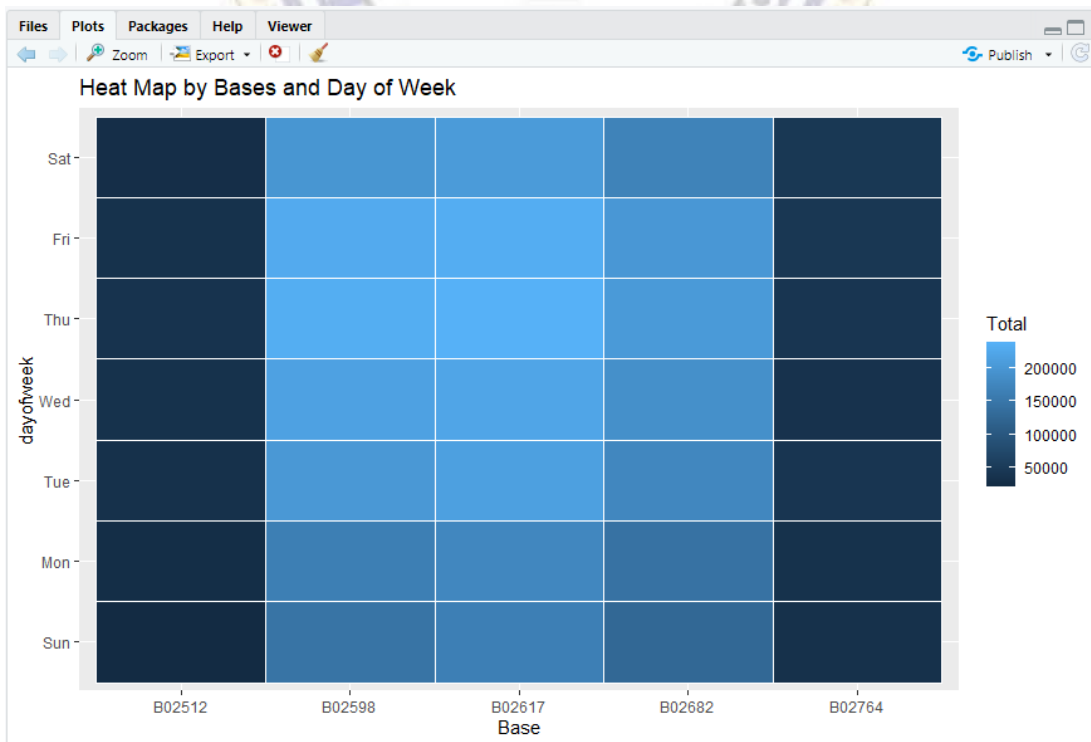
## Plotting the Heatmap of trips by month and day of week



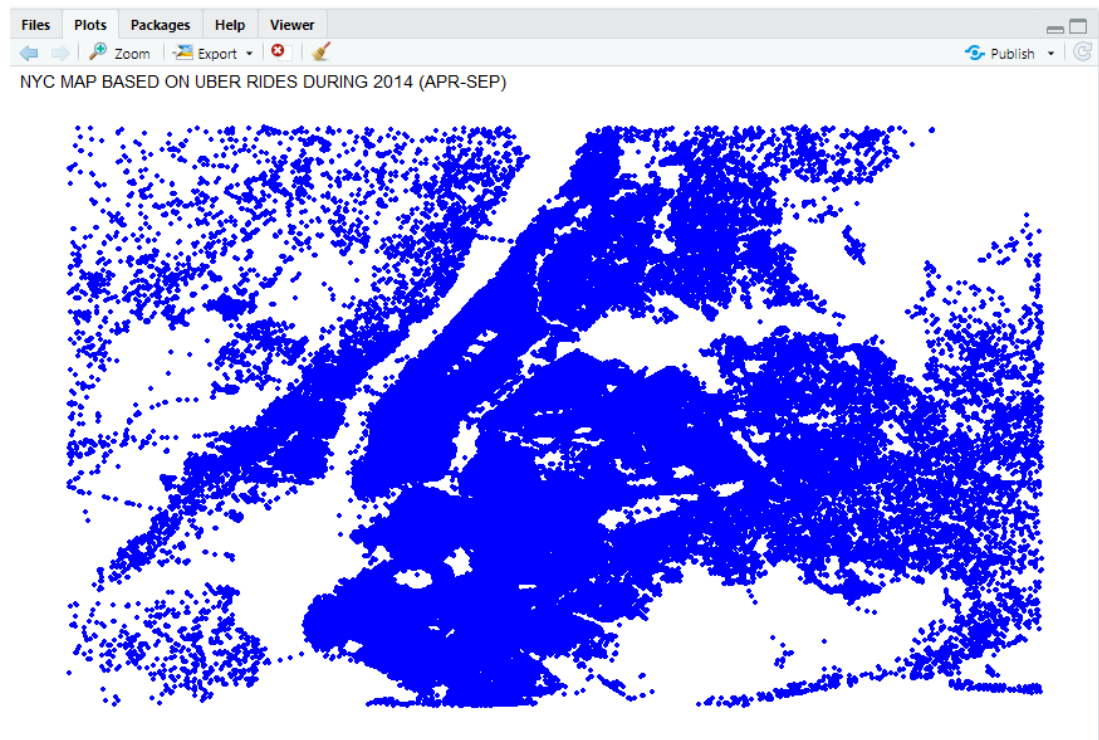
## Plotting a Heatmap visualization of trips by month and bases



## Plotting a Heatmap visualization of trips by bases and day of week



## Creating a map visualization of rides in New York





## References

## REFERENCES

- [1] Buja, Andreas, et al. "Data Visualization with Multidimensional Scaling." *Journal of Computational and Graphical Statistics*, vol. 17, no. 2, 2008, pp. 444–472.
- [2] Rossini, A. J., et al. "Simple Parallel Statistical Computing in R." *Journal of Computational and Graphical Statistics*, vol. 16, no. 2, 2007, pp. 399–420. JSTOR, [www.jstor.org/stable/27594249](http://www.jstor.org/stable/27594249).
- [3] Buja, Andreas, et al. "Interactive High-Dimensional Data Visualization." *Journal of Computational and Graphical Statistics*, vol. 5, no. 1, 1996, pp. 78–99. JSTOR, [www.jstor.org/stable/1390754](http://www.jstor.org/stable/1390754).
- [4] Ihaka, Ross, and Robert Gentleman. "R: A Language for Data Analysis and Graphics." *Journal of Computational and Graphical Statistics*, vol. 5, no. 3, 1996, pp. 299–314.
- [5] Nickerson, David W., and Todd Rogers. "Political Campaigns and Big Data." *The Journal of Economic Perspectives*, vol. 28, no. 2, 2014, pp. 51–73.
- [6] Fox, Peter, and James Hendler. "Changing the Equation on Scientific Data Visualization." *Science*, vol. 331, no. 6018, 2011, pp. 705–708.
- [7] Maltese, Adam V., et al. "Data Visualization Literacy: Investigating Data Interpretation Along the Novice—Expert Continuum." *Journal of College Science Teaching*, vol. 45, no. 1, 2015, pp. 84–90.
- [8] Zacharoula Papamitsiou, and Anastasios A. Economides. "Learning Analytics and Educational Data Mining in Practice: A Systematic Literature Review of Empirical Evidence." *Journal of Educational Technology & Society*, vol. 17, no. 4, 2014, pp. 49–64.
- [9] Nagaraju, V, et al. "An Open-Source Tool to Support the Quantitative Assessment of Cyber Security for Software Intensive System Acquisition." *Journal of Information Warfare*, vol. 16, no. 3, 2017, pp. 31–50.
- [10] Y. Demchenko et al., "EDISON Data Science Framework: A Foundation for Building Data Science Profession for Research and Industry," 2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), 2016, pp. 620-626, doi: 10.1109/CloudCom.2016.0107.
- [11] J. S. Saltz and N. W. Grady, "The ambiguity of data science team roles and the need for a data science workforce framework," 2017 IEEE International Conference on Big Data (Big Data), 2017, pp. 2355-2361, doi: 10.1109/BigData.2017.8258190

- [12] Y. Demchenko, A. Belloum, C. de Laat, C. Loomis, T. Wiktorski and E. Spekschoor, "Customisable Data Science Educational Environment: From Competences Management and Curriculum Design to Virtual Labs On-Demand," 2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), 2017, pp. 363-368, doi: 10.1109/CloudCom.2017.59.
- [13] A. Gunawan, M. L. Fong Cheong and J. Poh, "An Essential Applied Statistical Analysis Course using RStudio with Project-Based Learning for Data Science," 2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE), 2018, pp. 581-588, doi: 10.1109/TALE.2018.8615145.
- [14] "IEEE Standard Hardware Description Language Based on the Verilog(R) Hardware Description Language," in IEEE Std 1364-1995 , vol., no., pp.1-688, 14 Oct. 1996, doi: 10.1109/IEEESTD.1996.81542.
- [15] D. Wang, Y. Ji, B. Bai and Y. Shi, "Practice of ideological and political education in the course of “R language foundation”," 2020 International Conference on Information Science and Education (ICISE-IE), 2020, pp. 75-78, doi: 10.1109/ICISE51755.2020.00023
- [16] Ihaka, Ross, and Robert Gentleman. “R: A Language for Data Analysis and Graphics.” Journal of Computational and Graphical Statistics, vol. 5, no. 3, 1996, pp. 299–314.
- [17] Nagaraju, V, et al. “An Open-Source Tool to Support the Quantitative Assessment of Cyber Security for Software Intensive System Acquisition.” Journal of Information Warfare, vol. 16, no. 3, 2017, pp. 31–50.
- [18] A. M. Olney and S. D. Fleming, "A Cognitive Load Perspective on the Design of Blocks Languages for Data Science," 2019 IEEE Blocks and Beyond Workshop (B&B), 2019, pp. 95-97, doi: 10.1109/BB48857.2019.8941224
- [19] L. von Rueden et al., "Informed Machine Learning - A Taxonomy and Survey of Integrating Prior Knowledge into Learning Systems," in IEEE Transactions on Knowledge and Data Engineering, doi: 10.1109/TKDE.2021.3079836
- [20] Yi Tan and Guo-Ji Zhang, "The application of machine learning algorithm in underwriting process," 2005 International Conference on Machine Learning and Cybernetics, 2005, pp. 3523-3527 Vol. 6, doi: 10.1109/ICMLC.2005.1527552
- [21] M. Ross, C. A. Graves, J. W. Campbell and J. H. Kim, "Using Support Vector Machines to Classify Student Attentiveness for the Development of Personalized Learning

Systems," 2013 12th International Conference on Machine Learning and Applications, 2013, pp. 325-328, doi: 10.1109/ICMLA.2013.66

[22] Nickerson, David W., and Todd Rogers. "Political Campaigns and Big Data." The Journal of Economic Perspectives, vol. 28, no. 2, 2014, pp. 51–73.

[23] J. S. Saltz and N. W. Grady, "The ambiguity of data science team roles and the need for a data science workforce framework," 2017 IEEE International Conference on Big Data (Big Data), 2017, pp. 2355-2361, doi: 10.1109/BigData.2017.8258190

[24] D. Wang, Y. Ji, B. Bai and Y. Shi, "Practice of ideological and political education in the course of "R language foundation"," 2020 International Conference on Information Science and Education (ICISE-IE), 2020, pp. 75-78, doi: 10.1109/ICISE51755.2020.00023

